Protocol Analysis Simulation System PASS-1000 for MIL-STD-1553/1773 User's Manual

SBS Technologies, Inc.
2400 Louisiana Boulevard, NE
AFC Building 5, Suite 600
Albuquerque, NM 87110-4316
Fax: 505-875-0400
Email: sbshelp@sbs.com
http://www.sbs.com
http://www.resource.sbs.com
800-SBS-1553 or 505-875-0600

Applies to models: ABI-PMC2, ASF PMC2, ABI-PMC2, -XT, ASF PMC2-XT, ABIPMC2-CC, ASF PMC2-CC

Doc PN: 501-553801-00

Table of Contents

1: Introducing the PASS 1000 1-1

The PASS-1000 Advantage 1-2
Manual Organization 1-3
Conventions 1-4
Features of the PASS-1000 1-6
Configuration Requirements 1-7
The Package 1-7
Customer Support Service 1-8

2: Installation 2-1

Installation Process Overview 2-2
Required Connectors 2-3
Unpacking and Inspecting the Equipment 2-4
Installing the PASS-1000 Software 2-5
Allocating System Resources 2-8
Editing the PASS Configuration File 2-18
Installing the PASS Card into the PC System 2-31
Installing the OS Specific Device Drivers 2-44
Miscellaneous Connection Issues 2-51
Adding the True-Type Fonts 2-65
Testing Your Installation 2-66
Uninstalling the PASS-1000 Software 2-74
Disabling Adobe Type Manager 2-75

3: User's Guide 3-1

Definition of Terms 3-2
PASS-1000 Basics 3-2
Bus Controller Simulation 3-14
Remote Terminal Simulation 3-30
Bus Monitoring 3-41
Data Logging 3-63
Playback 3-74
Utilities 3-76

A: Error Reporting A-1

Word Errors A-2
Message Errors A-3
Error Codes in Count Analysis A-4

B: ASCII File Formats B-1

BC List B-1
Example of BC List File B-6
RT Setup B-8
Example of RT Setup File B-10
Misc. Setup B-11
Example Of Misc. Setup File B-16

C: Dynamic Link Library Calls C-1

PASSLIB.H Functions C-1
PASSLIB2.H Functions C-16
Example Program C-30

1: Introducing the PASS 1000

Welcome to the Protocol Analysis and Simulation System (PASS-1000): the most advanced avionics network analysis system available. The PASS-1000 operates on an IBM personal computer (PC) or compatible with a 486/Pentium microprocessor running standard Microsoft Windows 95, or Windows NT. The PASS-1000 system consists of an SBS ABI-PASS module, advanced windows software distribution, and this user's manual.

The PASS-1000 performs simultaneous, independent simulation of full bus controller (BC), 31 remote terminals (RT), and bus monitoring functions. The system provides an intuitive user interface and advanced features such as data logging to disk, reconstructive playback, percentage bus loading and activity displays, error injection and detection, and DLL libraries for relaying real-time data to other Windows based data analysis packages.

1.1 The PASS-1000 Advantage

The task of developing a system requiring communication over a MIL-STD-1553 bus is often quite complex. The MIL-STD-1553 interface in itself is very software intensive requiring significant investments in design, test, and maintenance. Throughout this life-cycle, the engineer relies upon test equipment such as a bus analyzer to verify proper communication, explore problems, and record data. For many of these bus analyzers, a great amount of setup time is required to make bus connections and type in the configuration necessary to extract meaningful data. Once the data is captured, it is often difficult to use and analyze. For any MIL-STD-1553 system, the PASS-1000 not only provides a simple means to view and record bus activity, but also offers several avenues for data analysis. It has been designed specifically for capturing and displaying data with no complicated initialization sequences. Because the PASS-1000 operates under Windows, a few clicks of the mouse button is all it takes to emulate a remote terminal and execute bus controller commands. Three additional clicks will monitor all traffic! The graphical interface and open architecture features of the PASS-1000 make it a superior alternative to the clumsy-to-use, closed system analyzers familiar to the industry.

The PASS-1000 is effective in a wide range of data analysis applications. It can be used with engineering analysis packages like LabVIEW for Windows to view and control bus data using a virtual instrument. Since the PASS-1000 supports DLL shared libraries, other client software may be used, such as analysis packages or application software written for Windows, to exchange real-time data with the PASS-1000. One unique feature of the system is its ability to archive data to a variety of devices. Depending on system and network performance, data may be written in real-time to disk, tape, a network like Ethernet, or another device that best suits the user's needs. The data may be read from the device to view the events that occurred on the bus or analyzed using the user's own tools. The format of the recorded data is provided, and software support is available upon request.

The PASS-1000 is useful during the entire lifecycle of a system. It is invaluable during development and design phases to verify hardware and software problems. It is useful for detecting or injecting Parity, Sync, Gap, Word Count, Bit Count, Illegal Mode Codes, and No Response errors to test the fault tolerance of remote terminals and the bus controller. Later during operational phases, the PASS-1000 can be used when more data collection and analysis tasks are required.

1.2 Manual Organization

This manual provides information on the use of the PASS-1000 window software. It is organized into the following main sections:

- Introduction
- Installation
- User's Guide
- Appendices

Introduction The Introduction introduces the PASS-1000 and summarizes its capabilities.

Installation The Installation chapter describes the steps necessary to configure and install the PASS-1000 software into the host system. (If the computer option was purchased, the PASS-1000 has already been installed.)

User's Guide The User's Guide consists of information needed to get the PASS-1000 up and running quickly. Most of this section is identical to the software on-line help. It explains how to use all PASS-1000 functions.

Appendices Several appendices follow the User's Guide:

- Appendix A describes Error Reporting.
- Appendix B contains the IRIG installation and user's guide (Not required for the PC3, PCI, and PCMCIA versions of the PASS-1000.)
- Appendix C describes the format of PASS ASCII files.
- Appendix D provides information about DLL calls for single PASS devices.
- Appendix E provides information about DLL calls for multiple PASS devices.
- Appendix F gives 16/32 bit Thunk build instructions
- Appendix G contains information about general troubleshooting and diagnostics

The SBS document *An Interpretation of MIL-STD-1553* is included with PASS shipments to assist the PASS-1000 user with 1553 protocol.

1.3 Conventions

The following conventions appear in this document. These conventions may differ from those used in other SBS publications. The sections listed below describe each convention in more detail:

- ➢ Typographic Conventions
- ➢ Words Having Special Meaning
- ➢ Compound Keystrokes and Menu Selections
- ➢ Symbols

1.3.1 Typographic Conventions

Table 1.3.1 shows the typographic conventions used in this document.

Table 1.3.1: Typographic Conventions

Element	Use in Body Text	Use in Procedures
Italic	➢ Document, chapter, section, and topic titles and cross references. ➢ Emphasis.	➢ Filenames and directory paths.
Bold	➢ (Not used in body text.)	➢ Controls, dialogs, menus, and text or numeric fields that appear on the screen. ➢ Keys on your keyboard.
`Courier Roman`	➢ Code examples.	➢ Simulating the appearance of screens.
`Courier Bold`	➢ Library function calls and syntax. ➢ Emphasizing lines of code.	➢ Commands and other information that you type as given.
Angle brackets, e.g.,< >	➢ Enclosing variable information that you type (without the brackets) in place of a dummy variable.	➢ Enclosing variable information that you type (without the brackets) in place of a dummy variable.

The point size of the text varies depending on whether it is used in body text, code examples, notes, screens, or procedures.

1.3.2 Words Having Special Meaning

In procedures, the words “Enter” (or “enter”) and “Type” (or “type”) have special meanings that are indicated in Table 1.3.2.

Table 1.3.2: Words with Special Meaning

Word	What It Means
Enter	Key in the specified text or variable information and press the **Return** key.
Type	Key in the specified text. Do not press **Return**.

1.3.3 Compound Keystrokes and Menu Selections

Compound Keystrokes

➢ Whenever a procedure instructs you to press multiple keys, the keys appear separated by a double angle bracket "»". An example appears in Table 1.3.3.

Menu Selections

➢ Whenever a procedure instructs you to select an item from a pull-down menu, the menu items appear separated by a double angle bracket "»". An example is shown in Table 1.3.3.

Table 1.3.3: Examples of Compound Keystrokes and Menu Selections

Instruction	What It Means
Press **Ctrl** » **Alt** » **Delete**.	Press the **Ctrl**, **Alt**, and **Delete** keys simultaneously.
Select **File** » **Open**.	Select **Open** from the **File** menu.

1.3.4 Symbols

The following symbols appear throughout this manual.

⚠	**Warning:** Paragraphs next to this symbol contain information critical to product operation, or to your safety.
✋	**Note:** Paragraphs next to this symbol contain information important to product operation.
💡	**Tip:** Paragraphs next to this symbol contain useful tips.
📖	**Cross Reference:** Paragraphs next to this symbol contain cross references to the *ABI/ASF User's Manual* or to a related page in a *Getting Started* manual.

1.4 Features of the PASS-1000

The PASS-1000 includes the features most requested by engineers and system integrators:

1.4.1 MIL-STD-1553 Interface

- ➢ Simulates full BC activity.
- ➢ Simulates full 31 RTs.
- ➢ 256 kilobytes of onboard memory.
- ➢ Selective Monitoring Achieved Through Large, Dual Buffers.
- ➢ 250 nanosecond Clock Resolution for measuring Intermessage Gap and Response Times.
- ➢ Digital and Analog Terminal Activity Displays.
- ➢ Bus Loading Displays.
- ➢ Injection and detection of Parity, Sync, Gap, Word Count, Bit Count, and other Protocol Errors.
- ➢ Supports One or more Dual-Redundant Buses.
- ➢ ABI-PASS board running a 10 MIP DSP processor and 256 kilobytes RAM.

1.4.2 Open System Architecture

- ➢ Pentium PC-based portable, desktop, and rack mount systems.
- ➢ Microsoft Windows 3.x, 95, or NT User Interface. Intuitive window layouts for easy use.

1.4.3 Archiving and Engineering Analysis

- ➢ Archives 100% of the Bus Traffic. Multiple stream archiving depends on pc system capabilities and bus loading. Advanced playback display and search options.
- ➢ Archives to disk, tape, and DMA. Specific device drivers are available upon request.
- ➢ Supports engineering analysis packages through standard DLL. Data analysis, EU conversions, and graphics display packages with drivers available upon request.

1.5 Configuration Requirements

- ➢ IBM-PC compatible Pentium system (portable, desk top, or rack mount).
- ➢ Minimum of 16 megabytes of RAM.
- ➢ Minimum of 50 megabytes of space on the hard disk drive. (The type of hard disk and controller may affect performance during data logging. High performance disk drives are recommended.)
- ➢ High resolution color VGA or Super VGA monitor.
- ➢ Microsoft Windows 3.1, 95, and NT compatible.
- ➢ SBS Technologies ABI-PASS MIL-STD-1553 interface board.
- ➢ SBS Technologies PASS-1000 windows software. (Installed in SBS PC Systems)

1.6 The Package

The PASS-1000 system consists of the following items:

- ➢ PC compatible Pentium system (optional).
- ➢ *One or more ABI-PASS boards.
- ➢ *PASS-1000 software diskettes.
- ➢ PASS-1000 User's Manual (this manual).
- ➢ Cable Assembly (except 1773).
- ➢ Two terminators (for each channel on the ABI-PASS).

* These items have already been installed and configured in the PC if the complete PASS-1000 system was purchased.

1.7 Customer Support Service

SBS Technologies is dedicated to providing technically superior products and the best customer support possible. Full support for the PASS-1000 product is provided, including any reasonable assistance with the entire integration effort. For assistance, please contact SBS Technologies via phone, fax, regular mail, or e-mail.

SBS Technologies, Inc.
2400 Louisiana Boulevard NE, Building 5, Suite 600
Albuquerque, New Mexico 87110

1-877-TECHSBS or 505-875-0600
FAX (505) 875-0400

e-mail: *sbshelp@sbs.com*

In addition, you may obtain the ABI/ASF User's Manual and the ABI/ASF C Libraries at no cost by contacting SBS Technologies at the telephone numbers listed above.

1.7.1 Your Feedback is Important

Comments and suggestions about SBS Technologies documentation and products are appreciated. You can use the "Change Request" form provided on the following page to send your comments to SBS Avionics Technologies via FAX or mail.

PASS-1000 Software or Manual Change Request	
SBS Technologies requests your input to keep our software and manuals current. If an error is found or a modification is requested, please fill out the following information and fax or mail us the request. We will try to implement your request in the next revision and will send you change pages to update your manual set. Thank you.	
Name:	
Company:	
Address:	
City/State/Zip:	
Phone:	FAX:
Manual Change	
Name of Manual:	Section:
Please record the page numbers and describe the error or change:	

PASS-1000 Software or Manual Change Request

PASS 1000 Software Change
Please describe the error or change:
FAX To: 1553 Product Manager at 505-875-0400 Mail To: 1553 Product Manager 2400 Louisiana Boulevard NE Building AFC5, Suite 600 Albuquerque, NM 87110 Address general questions or comments to: 1553 Product Manager 1-877-TECHSBS or 505-875-0600

2: Installation

The sections in this chapter describe how to unpack, configure, and install the PC2, PC3, cPCI, PCI, PCM, PCM2, 1773 PCI, PC104, and FW5000 versions of the Protocol Analysis and Simulation System (PASS-1000) into the host PC system. The specific sections are:

- ➢ Installation Process Overview
- ➢ Required Connectors
- ➢ Unpacking and Inspecting the Equipment
- ➢ Installing the PASS-1000 Software
- ➢ Allocating System Resources
- ➢ Editing the PASS Configuration File
- ➢ Installing the PASS Card into the PC System
- ➢ Installing the OS Specific Device Drivers
- ➢ Miscellaneous Connection Issues
- ➢ Adding the True-Type Fonts
- ➢ Testing Your Installation
- ➢ Uninstalling the PASS-1000 Software
- ➢ Disabling Adobe Type Manager

2.1 Installation Process Overview

The process of installing the PASS card and the PASS application varies according to the type of card you have and the operating system you want to run it under. The general installation sequence appears below.

Warning: SBS highly recommends the installation sequence below. This sequence is independent of PC operating system, PASS model, and PASS media distribution (Floppy Diskettes, CDROM, or SBS FTP Site).

1. Installing the PASS software (full or upgrade version). Requires rebooting your computer.
2. Determine and reserve system resources for the PASS card if necessary, then edit the *pass.cfg* file. May require rebooting your computer. The table below describes the possibilities.

Card:	Window 95/98	Windows NT
PC2	Determine resources Reserve resources Edit *pass.cfg* file	(Not supported)
PC3	Determine resources Reserve resources Edit *pass.cfg* file	Determine resources Edit *pass.cfg* file
cPCI	Edit *pass.cfg* file	Edit *pass.cfg* file
PCI/FW 5000	Edit *pass.cfg* file	Edit *pass.cfg* file
PCM	Edit *pass.cfg* file	(Not supported)
PCM2	Edit *pass.cfg* file	(Not supported)
1773 PCI	Edit *pass.cfg* file	Edit *pass.cfg* file
PC 104	Determine resources Reserve resources Edit *pass.cfg* file	Determine resources Edit *pass.cfg* file

3. Installing the PASS card(s). Requires rebooting your computer.
4. Installing PASS software device drivers. Depending on the PASS model and PC operating system, this step may not be required. May require rebooting your computer.
5. Testing and verifying your installation.

If the PASS board and PASS-1000 software were purchased separately from the PC, please follow these instructions carefully to install these items into your portable, desktop, or rack mount PC. If the complete PASS-1000 system was purchased, including the PC, the system is fully installed and ready for use. See *Chapter 3: User's Guide* for operating instructions.

2.2 Required Connectors

One cable assembly, part number CA2088 (single channel) or CA2087 (dual channel), is provided with the PC3 and PCI. This assembly attaches to the J1 connector on the PC3 and PCI panels. It provides cable leads for making connections to Bus A and Bus B as well as a cable lead with a DB15 connector for making auxiliary connections. It is available from Milestek (see address below). Table 2.2.1 lists the mating connectors required for the cable assembly leads.

Table 2.2.1: Mating Connectors

Card Type	Cable Assembly Leads	Part Number of Mating Connector	Manufacturer
PC3, PCI	Bus A lead(s)	PL75	Trompeter
	Bus B lead(s)	PL75	Trompeter
	DB15	low-density 15-pin plug connector	
PCMCIA		PL75	Trompeter
PC104	(connector only)	746285	Amphenol
FW5000	Bus A lead(s)	PL75	Trompeter
	Bus B lead(s)	PL75	Trompeter
	DB15	low-density 15-pin plug connector	
PCI-1773	1773	FCA866-3M	Alpha Wire Corporation

SBS sells bus configurations, part numbers BUS-2 (2-stub coupler) and BUS-3 (3-stub coupler), for transformer coupled systems. Contact SBS at 1-877-TECHSBS for more information.

Other components are available from:
MilesTek
1 Lake Trail Drive
Argyle, Texas 76226
Attn: Frank or Jeanette Miles

1-800-524-7444 or (940) 484-9400
Fax: (940) 484-9402

MilesTek also provides the following:

Item	Part Number
Twin-axial cable	CA2009-xxx (where xxx denotes the length in inches)
1553 bus terminator	10-06403-025

2.3 Unpacking and Inspecting the Equipment

The PASS board should be carefully unpacked and inspected for any visible damage that could have occurred during shipment. If there is visible damage, contact the Maintenance/Repairs/Warranty department at SBS at 877-TECHSBS and have the board and its serial number available.

Warning: This is an electronic product that is sensitive to electrostatic discharge. Normal precautions should be observed in handling the board to prevent damage.

2.4 Installing the PASS-1000 Software

Warning: Prior to upgrading to a newer version of PASS software, there should only be one copy of PASS 1000 and one copy of *passlib.dll* installed on all hard drives in your system.

Warning: If you currently have a version of the PASS software installed and have received a new distribution, be sure to verify that the distribution is labeled *Upgrade* or contains an upgrade folder. If it does not, contact SBS to obtain the upgrade version. Reinstalling the installation version of PASS-1000 Software over an existing version (instead of installing an upgrade version) may cause unpredictable results.

PASS Distribution Media Formats

PASS-1000 is distributed in several formats.Table 2.4.1 describes each format.

Table 2.4.1: PASS-1000 Distribution Formats

Media:	Description:
Floppy Diskette	Full version Diskettes are labeled “PASS 1000 Installation Disk Version x.xx Disk x of x”. Upgrade version diskettes are labeled “PASS 1000 Update Disk Version x.xx Disk x of x”. For both versions, the first diskette contains the installation program, *setup.exe*. Subsequent disks contain the rest of the installation.
CDROM	CDROM labeled “SBS Technologies Inc. PASS-1000 Version x.xx”. The CD contains two folders, FULL and UPGRADE. Each of these folders contains the single file installation program, *setupexe.exe*. This one file contains the entire installation file set.
SBS FTP Site	The directory */pub/passfiles* contains the full, *pass_Ver_x_xxx*, and upgrade, *pass_Ver_x_xxx_update*, sub directories. Each of these directories contains the single file installation program, *setupexe.exe*. This one file contains the entire installation file set.

Procedure

Follow the procedure below to install the PASS software.

1. To load your particular media distribution into your computer, see the instructions in Table 2.4.2.

Table 2.4.2: Media Loading Instructions

If you have:	What to do:
Floppy Diskette	Place diskettes labeled “PASS 1000 Installation Disk Version x.xx Disk 1 of x” into drive (a or b).
CDROM	Place the CD into the CD ROM drive.

If you have:	What to do:
SBS FTP Site	Create a temporary directory, e.g., c:\temp. Then place the file you obtained from the FTP site inside the temporary directory.

2. Each operating system uses a slightly different method to locate the installation program, refer to the table below for the correct method for your OS.

Table 2.4.3: Installation Program Startup

If your operating system is:	Then:
Windows 3.x or Windows NT 3.51	Select **Program Manager » File » Run**
Windows 95/98 or Windows NT 4.0	Select **Start » Run**

A dialog box appears asking you to locate the program you wish to run.

3. To start the installation process, refer to the instructions in Table 2.4.4.

Table 2.4.4: Program Locations

If you have:	What to do:
Floppy Diskette	Enter **`a:\setup`** (or **`b:\setup`** if the floppy is in the b: drive)
CDROM	To install the full version, enter **`<x>:\Full\setupex.exe`** To install the upgrade version, enter **`<x>:\Upgrade\setupex.exe`** where <**`x`**> is the CDROM drive letter.
SBS FTP Site	Enter **`<x>:\<path>\setupex.exe`** where <**`x`**> is the hard drive letter and <**`path`**> is the path to the directory containing the file downloaded from the FTP site.

4. Read the message and click **OK**.

5. By default, the program files will be installed in the c:\pass directory unless you enter a different path. After accepting the default, or entering another path, click **OK**.

6. A menu appears allowing you to select a complete or partial PASS installation. Select the desired installation level and click **OK**.

   The installation process begins. If you're installing from floppy disk, you will be prompted to insert the remaining floppies as needed.

7. A message window appears when the installation program has completed. Click **OK** to display the *readme.txt* file and, if installing from floppies, remove the last installation disk.

8. (Windows 3.1/95/98 only): *c:\autoexec.bat* contains the entry *Set PASS=c:\pass*.

   (Windows NT 4.0 only): Ensure environment variable PASS is set, for example, to *C:\PASS*. If the default directory differs from *C:\* then modify variable value accordingly. Check for the PASS definition, and if not present, click **Start » Settings » Controls Panel**, then double click the System icon. When the **System Properties** window opens, select the **Environment** tab. Click the **Variable** field and enter `Pass`. Then click the **Value** field and enter `c:\pass` and click **OK**.

Software Directory Structure

After the installation program *setup.exe* finishes, Table 2.4.5 describes where the software should be installed on the selected drive.

Table 2.4.5: Software Locations

Directory (As Installed)	Description
\pass	PASS software files, firmware (*.txt* and *.bit*) files, client (*sbs_clnt.exe*) file
\pass\dbase	Sample setup (*.ba3*) files, ASCII save (*.bc*, *.rt*, *.tl*, *.chn*, *.trn*, *.vew*) files, and example EXCEL (*.xls*) files.
\windows\system	The following DLL files: *bwcc.dll*, *owl252.dll*, *bids45.dll*, *bc450rtl.dll*, and *cw3215.dll*. (Borland Run Time Library files)
\pass\dev_drvr	PASS device driver files for Windows 95/98/NT.
\pass\passlib	Windows 95/98 *passlib.dll*, source code, examples, LabView 5.0 examples.
\pass\passlib.nt	Windows NT *passlib.dll*, source code, examples, LabView 5.0 examples.

2.5 Allocating System Resources

In order to use your PASS card, you need to make sure that PC system resources are made available to the PASS card. PASS cards need the following resources:

- ➢ Input/Output (I/O) Address Range
- ➢ Memory Address Range
- ➢ Interrupt Request Level (IRQ)

For some PASS card/OS combinations, you need only check that there are resources available, the card/OS combination automatically reserves what it needs. For other combinations, you will need to check *and* reserve resources. Table 2.5.1 describes all the combinations. Notice that not all combinations are supported.

Table 2.5.1: PASS Card Resource Requirements by Operating System

Card:	Windows 3.1	Windows 95/98	Windows NT
cPCI	(Not Supported)	*Check:* IRQ	*Check:* I/O Address Range Memory Address Range
PCI	(Not Supported)	*Check:* IRQ	*Check:* I/O Address Range Memory Address Range
FW 5000	(Not Supported)	*Check:* IRQ	*Check:* I/O Address Range Memory Address Range
1773 PCI	(Not Supported)	*Check:* IRQ	*Check:* I/O Address Range Memory Address Range
PC2	*Check and reserve:* IRQ I/O Address Range Memory Address Range	*Check and reserve:* IRQ I/O Address Range Memory Address Range	(Not Supported)
PC3	*Check and reserve:* IRQ I/O Address Range Memory Address Range	*Check and reserve:* IRQ I/O Address Range Memory Address Range	*Check:* I/O Address Range Memory Address Range
PC 104	*Check and reserve:* IRQ I/O Address Range Memory Address Range	*Check and reserve:* IRQ I/O Address Range Memory Address Range	*Check:* I/O Address Range Memory Address Range
PCM	*Check:* IRQ	*Check:* for 2 IRQs, 1 for the PASS card, 1 for Socket Services	(Not Supported)

Card:	Windows 3.1	Windows 95/98	Windows NT
PCM2	*Check:* IRQ	*Check:* for 2 IRQs, 1 for the PASS card, 1 for Socket Services	(Not Supported)

The subsections describe the possibilities:

➢ Allocating System Resources Under Windows 3.x
➢ Allocating System Resources Under Windows 95/98
➢ Checking System Resources Under Windows NT

Depending on your PASS card, you may have to perform all or just part of a procedure.

Warning: The process of reserving resources for the PASS card is potentially iterative and may require rebooting your system several times in order to find an acceptable set of values.

Warning: Always make a backup of your system files before altering any values.

Warning: For all operating systems, SBS PASS cards require unique, non-sharable addresses.

Warning: For Windows 3.x and Windows 95 systems, SBS PASS cards require unique, non-sharable interrupt values.

Note: Resource selection could be limited by other cards installed in your PC and the order in which these cards are seen by the hardware and by the operating system.

2.5.1 Allocating System Resources Under Windows 3.x

If you're using Windows 3.x, Table 2.5.2 describes the PASS cards you can use and the resources you need to check and, if necessary, reserve.

Table 2.5.2: PASS Card Resource Requirements for Windows 3.x

Card:	Windows 3.1
cPCI	(Not Supported)
PCI	(Not Supported)
FW 5000	(Not Supported)
1773 PCI	(Not Supported)
PC2	*Check and reserve:* IRQ, I/O Address Range, Memory Address Range
PC3	*Check and reserve:* IRQ, I/O Address Range, Memory Address Range
PC 104	*Check and reserve:* IRQ, I/O Address Range, Memory Address Range
PCM	*Check:* IRQ
PCM2	*Check:* IRQ

When you're using Windows 3.x, it's a little more difficult to locate resources than when using Windows 95/98, or NT. Windows 3.x comes with a DOS-level diagnostic program *Microsoft Diagnostics* (MSD). In the procedure below, you will be using MSD to check for system resources.

Checking for Resources under Windows 3.x

1. At the DOS prompt, enter **`msd`**.

   The msd program menu appears.

2. Use the **IRQ Status**, **Memory**, and **TSR** programs to identify a free IRQ, I/O Address Range, and Memory Address Range. Write these values down. You will need then later when you edit the *pass.cfg* file.

   If you're using a PCM or PCM2 card, the remaining Steps of this procedure should have been performed by your PCMCIA socket software (e.g., CardSoft's Card Wizard). Skip the rest of this procedure and go to Section 2.6, Editing the PASS Configuration File.

Reserving Resources under Windows 3.x

3. When you're using MS-DOS/Windows 3.x, you need to exclude memory from the operating system. To exclude memory from the operating system, use the Notepad application to add the following lines to the *config.sys* file:

   ```
   Device=c:\Dos\himem.sys

   Device=c:\Dos\Emm386.exe NoEms X=D000-Dfff
   ```

   Where the value of **`X`** is the range of memory you identified in Step 2 above.

4. Starting in column 1, add the line below to the [386EnH] section in the *system.ini* file. The file is located in the *c:\windows\system* directory.

```
emmexclude=D000-Dfff
```

Note: Address and IRQ settings cannot be reserved under Windows 3.1 or versions of MS-DOS 6.22 or older. Some system BIOS allow IRQ reservations via CMOS settings. If the CMOS in your system has this capability, then an area exists that allows you to specify whether an IRQ will be associated with the PCI or ISA (legacy) bus. Depending on the number of PASS cards in your system, specify one or more IRQ's to be legacy.

These changes will not take effect until you reboot your system.

5. Reboot your system.

6. Start Windows. If Windows fails to start, or the Program Manager is not visible, a resource conflict exists; usually in the memory address range, but sometimes in the I/O address range. Repeat this procedure, selecting a different set of resource values.

7. Once you determine an acceptable set of resources, proceed to Section 2.6, for instructions on editing the *pass.cfg* file to reflect the resources you allocated.

2.5.2 Allocating System Resources Under Windows 95/98

If you're using Windows 95/98, Table 2.5.3 describes the PASS cards you can use and the resources you need to check and, if necessary, reserve.

Warning: SBS has encountered problems with resource allocation when the card is installed directly after video acceleration cards. Video acceleration cards have been known to consume more resources than they report to Windows. To avoid these problems, move the PASS card to a resource region before, or significantly after, the video acceleration card.

Table 2.5.3: PASS Card Resource Requirements for Windows 95/98

Card:	Windows 95/98
cPCI	*Check:* IRQ
PCI	*Check:* IRQ
FW 5000	*Check:* IRQ
1773 PCI	*Check:* IRQ
PC2	*Check and reserve:* IRQ, I/O Address Range, Memory Address Range
PC3	*Check and reserve:* IRQ, I/O Address Range, Memory Address Range
PC 104	*Check and reserve:* IRQ, I/O Address Range, Memory Address Range
PCM	*Check:* for 2 IRQs
PCM2	*Check:* for 2 IRQs

Note: These parameters are system dependent, consult the computer system documentation to determine available (unused) values. If a resource is selected which is used by another device, such as a video card, data acquisition card, or telemetry card, you may have to manually select new resources, or undetermined results will occur.

Checking for Resources under Windows 95/98

To check and reserve the necessary resources, complete the following steps:

1. Select **Start** » **Settings** » **Control Panel**.

   The **Control Panel** window appears.

2. Double click the **System** icon.

   The **System Properties** window appears.

3. Determine the version of Windows installed on your machine by selecting **Start** » **Settings** » **Control Panel**, then opening the **System** icon.

   The **System Properties** Window appears. The Windows version appears in the upper right. Write the version down in case you need to contact SBS Technical support.

4. Click the **Device Manager** tab.

   A list of devices appears.

5. Click the **Computer** icon located at the top of the list, then click **Properties**.

   The **Computer Properties** window appears. The **View Resources** tab of this window provides information about Input/Output address ranges, memory address ranges, and IRQs.

6. Click the **Memory** radio button.

   A list of devices appears with their memory ranges (in hex).

7. Scan the list for a free memory address range. A free memory address range will be a range that *does not show up in the list*. Write this number down in a convenient place, you will need them later when you edit the *pass.cfg* file.

8. If the version of Windows installed on your computer is **4.00.950 B** or newer, skip to Step 16.

9. Select **Start** » **Programs** » **Accessories** » **Notepad**.

   The Notepad application opens.

10. Click **File** » **Open** and enter **`c:\config.sys`** into the **Open** dialog which appears.

    The **`config.sys`** file appears.

11. If the following two lines are not present in the file, add:

    ```
    Device=c:\windows\himem.sys

    Device=c:\windows\Emm386.exe ON X=D000-Dfff
    ```

    Where **`c:\windows`** is the Windows 95 install directory. and **`D000-DFFF`** is the region to exclude.

12. Save the file.

13. Click **File** » **Open** and enter **`c:\windows\system.ini`** into the **Open** dialog which appears.

    The **`system.ini`** file appears.

14. Place the cursor in the [386Enh] area and add the following statement if not already present:

```
emmexclude=D000-DFFF
```

Warning: Make sure you start in column 1 and that you are in the [386Enh] section. If you make an error, and save the file, your computer will likely crash when you restart. Should this occur, you can recover by starting Windows in *Safe Mode,* and editing the file correctly, then restarting.

15. Save the file, exit Notepad, and click on the **Control Panel** window to bring it to the front.

16. Click the **Interrupt** radio button and scan the list that appears for a free IRQ. If you have a PCM, or PCM2 card, check for *two* IRQs, one for the PASS card and another for Socket Services. A free IRQ will be a number that *does not show up in the list*. Write this number down in a convenient place, you will need then later when you edit the *pass.cfg* file. If you have a cPCI, PCI, FW 5000, 1773 PCI, PCM, or PCM 2 card, skip the rest of this procedure and go to Section 2.6, Editing the PASS Configuration File.

17. Click the **Input/output (I/O)** radio button.

    A list of devices appears with their I/O address ranges (in hex).

18. Scan the list for a free I/O address range. A free I/O address range will be a range that *does not show up in the list.* Single channel cards require a range of 8h while dual channel cards require a range of 10h. Write this number down in a convenient place, you will need then later when you edit the *pass.cfg* file.

19. Select the **Reserve Resources** tab, then click the **Add** button, and enter the desired interrupt request value. If the value is unused, it will be added to the **Setting** box.

20. Select the **I/O** button, then click the **Add** button, and enter the desired I/O address range. If the value is unused, it will be added to the **Setting** box.

21. Select the **Memory** tab, then click the **Add** button, and enter the desired memory address range. If the value is unused, it will be added to the **Setting** box.

22. Specify the interrupt level, address range, and memory range for each installed PASS device. Write down these values. You will need them when you edit the *pass.cfg* file.

23. Reboot the system to restart Windows 95.

Note: Regardless of PASS card type, you need to check system resources to insure that PASS card is not in conflict with other system resources and devices.

24. Check to see if the resource reservation was successful by clicking **Start » Settings » Control Panel**, then double clicking on the System icon.

The **System Properties** window appears.

25. Click on the **Device Manager** tab, then examine the resulting list of resources for yellow symbols containing exclamation (!) or question (?) marks. These symbols indicate a resource allocation problem. You can check if the device is working properly by double clicking on the device entry, then the **General** tab.

Note: If there is a conflict, you need to select another set of resource values for the PASS card. If after several attampts, you continue to encounter problems, see Subsection 2.11.1.

26. Once you determine an acceptable set of resources, proceed to Section 2.6, for instructions on editing the *pass.cfg* file to reflect the resources you allocated.

2.5.3 Checking System Resources Under Windows NT

If you're using Windows NT, Table 2.5.4 describes the PASS cards you can use and the resources you need to check. It is not necessary to reserve particular resources under Windows NT. However, for the PC3 and the PC 104 cards, you will need to make note of the system resources values. You will need these values when you edit the *pass.cfg* file.

Table 2.5.4: PASS Card Resource Requirements for Windows NT

Card:	Windows 95/98
cPCI	*Check:* I/O Address Range, Memory Address Range
PCI	*Check:* I/O Address Range, Memory Address Range
FW 5000	*Check:* I/O Address Range, Memory Address Range
1773 PCI	*Check:* I/O Address Range, Memory Address Range
PC2	(Not Supported)
PC3	*Check:* I/O Address Range, Memory Address Range
PC 104	*Check:* I/O Address Range, Memory Address Range
PCM	(Not Supported)
PCM2	(Not Supported)

To check system resources, follow the procedure below.

1. Select **Start** » **Programs** » **Administrative Tools (Common)** » **Windows NT Diagnostics**.

   The **Windows NT Diagnostics** window appears.

2. Click the **Resources** tab.

   A list of IRQs appears.

3. Use the **I/O Port** and **Memory** buttons to verify the existence of unused system resources.

4. If you have a PC3 or a PC 104 card, write down the values of the unused resources. You will need then when you edit the *pass.cfg* file to reflect the new resources. Proceed to Section 2.6, *Editing the PASS Configuration File*.

(Move to Editing pass.cfg section)

For these cards, modify the *pass.cfg* file by uncommenting the two lines for each device. The rest of the line remains unchanged with the exception of the device name. You can change the 7th through the 11th character of the device name.

(End of text to move)

2.6 Editing the PASS Configuration File

The device configuration file, *pass.cfg*, is an ASCII text file that contains information to enable one or more PASS devices. The PASS 1000 software parses the file to determine the device type, base I/O address, base RAM address, and device names for each device. PASS 1000 must be able to correctly read and verify the information for a device before further initialization can proceed. When installing a PASS card or modifying its base I/O address switch, edit this file to reflect that switch value. To edit the file, select the *Pass.cfg* icon. Change the file as required, save it, and then close the window.

Note: If you are upgrading PASS 1000, your old *pass.cfg* file will be left unchanged. The upgrade program writes the newest version of the *pass.cfg* file to *c:\pass directory* with the filename *newpass.cfg*.

The *pass.cfg* file contains two lines for each PASS device configured in the system. The first line defines the BC/RT simulation functions and the second line defines the monitor function.

2.6.1 Conventions

Conventions for the *pass.cfg* file appear below:

- The mother board device label must be followed by a comma and an asterisk, e.g. (, *).
- Commas must separate fields.
- Spaces are not permitted before the **`Device=`** keyword.
- Comment lines begin with a semicolon (;).
- Comments are not permitted on the same line as an active field.

Sample *pass.cfg* File

A sample *pass.cfg* file appears below. To enable one or more PASS devices, un-comment (remove the semicolon) for the appropriate pair of lines and edit the configuration fields as required.

```
;
; This is the PASS configuration file for both MIL-STD 1553 and
; ARINC 429 devices. To enable a device, uncomment the
; appropriate pair of lines for that device.
;
;
; ****************************************************************
; ****** MIL-STD 1553 CARDS ***************************************
; ****************************************************************
;
;
; ****** Demo 1553 ***********************************************
; Use to run the PASS software without a card
;
;Device=ARI,0,0,0,Demo1BC,*
;Device=ARI,0,0,0,Demo1Mon,
;
;
; ****** PC2 ****************************************************
;
; NOTE: 1) The PC2 can only be configured as large (64K) window.
;       2) The PC2 card is not supported under Windows NT.
;
; For High Memory:
;Device=ABI, 390, F0, 15, BRM, *
;Device=ABI, 391, F2, 15, MONITOR
;
; For DOS Memory:
;Device=ADI, 3A0, 0D, 12, BRM2,*
;Device=ADI, 3A1, 0D, 12, Monitor2,
;
;
; ****** PC3-1 (Single Channel) ********************************
;
; Windows 3.x/95/98 large memory window:
;Device=ADI3, 390, 0D, 12, PC3BRM,*
;Device=ADI3, 390, 0D, 12, PC3MON,
;
; Windows 3.x/95/98 small memory window:
;Device=SADI3, 390, D4, 12, PC3BRM,*
;Device=SADI3, 390, D4, 12, PC3MON,
;
; Windows NT small memory window:
;Device=SNT3, 340, D0, 5, PC3BRM,*
;Device=SNT3, 340, D0, 5, PC3MON,
;
;
; ****** PC3-2 (Dual Channel) **********************************
;
; NOTE: 1) For Windows 3.x/95/98 the first channel must be enabled
;          to run the second channel. For Windows NT either
;          channel may be enabled and run independently.
;       2) The second channel IO port is the first channel IO
;          port plus 6.
;       3) Large window (64K) is not supported for the PC-3-2.
;       4) Both channels use the same interrupt.
;
; Windows 3.x/95/98 Channel 1:
;Device=SA3C1, 390, D4, 11, BRMC1,*
;Device=SA3C1, 390, D4, 11, MONC1,
; Windows 3.x/95/98 Channel 2:
;Device=SA3C2, 396, D4, 11, BRMC2,*
;Device=SA3C2, 396, D4, 11, MONC2,
;
```

```
; Windows NT Channel 1:
;Device=NT3_1, 340, CC, 11, BRMC1,*
;Device=NT3_1, 340, CC, 11, MONC1,
; Windows NT Channel 2:
;Device=NT3_2, 346, CC, 11, BRMC2,*
;Device=NT3_2, 346, CC, 11, MONC2,
;
;
; ****** PCMCIA *************************************************
;
; PCMCIA Windows 3.x/95/98
;Device=SMCIA, 0, 0, 0, MCIABRM,*
;Device=SMCIA, 0, 0, 0, MCIAMON
;
; PCMCIA 2 Windows 3.x/95/98
;Device=SMCIA2, 0, 0, 0, MCIABRM,*
;Device=SMCIA2, 0, 0, 0, MCIAMON
;
;
; ****** PCI ****************************************************
;
; NOTE: The PCI is not supported under Windows 3.1.
;
; Windows 95/98:
;Device=APCI, 0, 0, 0, PCIBRM,*
;Device=APCI, 0, 0, 0, PCIMON
;
; Windows NT:
;Device=NTPCI, 0, 0, 0, PCIBRM,*
;Device=NTPCI, 0, 0, 0, PCIMON,
;
;
; ****** PCI-2 (Dual Channel)************************************
;
; Windows 95/98 channel 1:
;Device=APCI2_1, 0, 0, 0, PCI2BRM1,*
;Device=APCI2_1, 0, 0, 0, PCI2MON1
;
; Windows 95/98 channel 2:
;Device=APCI2_2, 0, 0, 0, PCI2BRM2,*
;Device=APCI2_2, 0, 0, 0, PCI2MON2
;
; Windows NT:
;Device=NTPC1, 0, 0, 0, PCIBRM,*
;Device=NTPC1, 0, 0, 0, PCIMON,
;
; Windows NT:
;
; NOTE: 1) For Windows NT the first channel must be enabled
;          to run the second channel.
;       2) Both channels use the same interrupt.
;
;Device=NTPC2, 0, 0, 0, PCIBRM,*
;Device=NTPC2, 0, 0, 0, PCIMON,
;
;
; ****** PC104 and FW5000 ***************************************
;
; NOTE: 1) PC104 and FW5000 are not supported under Windows NT.
;       2) Only the large memory window (64K) is supported.
;
; Windows 3.x/95/98:
;Device=PC104, 390, 0D, 12, BRM2,*
;Device=PC104, 390, 0D, 12, MON2,
;
;
; **************************************************************
; ****** 1773 CARDS ********************************************
; **************************************************************
```

```
;
; Windows 95/98:
;Device=A1773, 0, 0, 0, PCIBRM,*
;Device=A1773, 0, 0, 0, PCIMON
;
; Windows NT:
;Device=NT1773, 0, 0, 0, PCIBRM,*
;Device=NT1773, 0, 0, 0, PCIMON
;
;
; ***************************************************************
; ****** ARINC 429 CARDS ****************************************
; ***************************************************************
;
;
; ****** ARINC 429 Demo *****************************************
; Use to run the PASS software without a card
;
;Device=R429, 0, 0, 0, DEMO-T/R,R=4,T=4,*
;Device=R429, 0, 0, 0, DEMO-MON
;
;
; ****** ARINC 429 PC8 (8 Channels) *****************************
;
; Windows 3.x/95/98:
;Device=D429, 390, 0, 10, 429T/R,R=4,T=4,*
;Device=D429, 390, 0, 10, 429Mon,
;
; Windows NT:
;Device=NT429A, 390, 0, 5, 429T/R,R=4,T=4,*
;Device=NT429A, 390, 0, 5, 429Mon,
;
;
; ****** ARINC 429 PC16 (16 Channels) ***************************
;
; NOTE: 1) The 16 channel card is configured as two 8 channel
;          devices with types L429A and L429B for Windows 3.x/95/98
;          and NT429A and NT429B for Windows NT.
;       2) Either device can be configured and run independently.
;       3) The D429 device type cannot be used because the
;          firmware is different.
;       4) The L429B (NT429B) device's IO port is set 8 above
;          the L429A (NT429A) device whose IO port is set via
;          the DIP switches on the board.
;       5) Both devices A and B use the same Interrupt.
;
; Windows 3.x/95/98 Device A:
;Device=L429A, 390, 0, 10, A429T/R,R=4,T=4,*
;Device=L429A, 390, 0, 10, A429Mon,
; Windows 3.x/95/98 Device B:
;Device=L429B, 398, 0, 10, B429T/R,R=4,T=4,*
;Device=L429B, 398, 0, 10, B429Mon,
;
; Windows NT Device A:
;Device=NT429A, 390, 0, 10, A429T/R,R=4,T=4,*
;Device=NT429A, 390, 0, 10, A429Mon,
; Windows NT Device B:
;Device=NT429B, 398, 0, 10, B429T/R,R=4,T=4,*
;Device=NT429B, 398, 0, 10, B429Mon,
;
;
; ****** ARINC 429 PCMCIA Card **********************************
;
; NOTE: 1) This product has been discontinued.
;       2) This sets channels 1,2,4,5,6, and 8 to 100 kHz and
;          channels 3 and 7 to 12.5 kHz.
;
;Device=PM429, 0, 0, 0, PM429T/R,R=4,T=4,HHLHHHLH*
;Device=PM429, 0, 0, 0, PM429Mon,
;
;
```

Note: The values for the Base I/O address, RAM base address, and the Interrupt level used here are for illustrative purposes only. The actual values in your *pass.cfg* file may be different. You may need to adjust these values to reflect the specific configuration of your system.

The general format of the two lines required for each device appears below:

```
DEVICE=device type, Base I/O Address, RAM Base Address, Interrupt level, de-
vice name, optional field, *

DEVICE=device type, Base I/O Address, RAM Base Address, Interrupt level, Mon-
itor name, optional field
```

Example 1 An example of these two lines in an actual *pass.cfg* file appears below:

```
DEVICE=SADI3, 390, D0, 15, PC3BRM, I, *
DEVICE=SADI3, 390, D0, 15, PC3MON, I
```

In this example, the device type is **`SADI3`**, a PC3 installed in a small DOS memory window, with base I/O address of **`390h`**, a RAM address of **`D0 0000h`**, and interrupt level of **`15`**.The PASS BC/RT simulation functions are named **`PC3BRM`** and the monitor function is named **`PC3MON.`** The optional field specifies the IRIG time source option.

Example 2 The following lines of code describe installation of a single and dual channel PCI cards on the same PCI bus under Windows NT 4.0. On that bus, the single channel card comes before the dual channel card.

Note: In most cases, computer manufacturers designate the slot closest to the CPU as Slot 1, the next closest as Slot 2, etc.

```
; ****** PCI *****************************************************
;
; NOTE: The PCI is not supported under Windows 3.1.
;       For Win95/98 systems with non-PASS SBS PCI cards,
;       the IO Port Field can be used to designate the
;       SBS Device number. Device Numbers are based on
;       PCI slots which are numbered on the mother board.
;       The SBS card with the lowest slot number is card 0.
;       If 0 is used for all IO ports the cards are selected
;       in the order they are listed in this file.
;
; Windows 95/98:
;Device=APCI, 0, 0, 0, PCIBRM,*
;Device=APCI, 0, 0, 0, PCIMON
;
; Windows NT:
Device=NTPC1, 0, 0, 0, PCIBRM,*
Device=NTPC1, 0, 0, 0, PCIMON,
;
;
; ****** PCI-2 (Dual Channel) **********************************
;
;
; Windows 95/98 channel 1:
;Device=APCI2_1, 0, 0, 0, PCI2BRM1,*
;Device=APCI2_1, 0, 0, 0, PCI2MON1
;
; Windows 95/98 channel 2:
;Device=APCI2_2, 0, 0, 0, PCI2BRM2,*
;Device=APCI2_2, 0, 0, 0, PCI2MON2
;
; Windows NT:
Device=NTPC1, 0, 0, 0, PCIBRM,*
Device=NTPC1, 0, 0, 0, PCIMON,
;
; Windows NT:
;
; NOTE: 1) For Windows NT the first channel must be enabled
;          to run the second channel.
;       2) Both channels use the same interrupt.
;
Device=NTPC2, 0, 0, 0, PCIBRM,*
Device=NTPC2, 0, 0, 0, PCIMON,
;
;
```

2.6.2 Device Type Keyword

Each line begins with a keyword named **DEVICE=** that specifies the type of board in the system and how it is mapped within memory. Within PC memory, the PASS can be memory mapped in one of the following ways:

- ➢ DOS Memory Area - Large Window (000000h - 0FFFFFh)
- ➢ DOS Memory Area - Small Window (000000h - 0FFFFFh)

In Large Window DOS Memory Mode, the PASS appears as a 64-kilobyte memory mapped window. In Small Window DOS Memory Mode, the PASS appears as a 16-kilobyte memory mapped window.

Note:
The ABI-PC2, the ABI-PC104, and the ABI-FW5000 can be configured only in large memory window mode.

The dual channel PC3 (ABI-PC3-2) can be configured only in small memory window mode.

For most applications, SBS recommends using the small memory window configuration.

The **DEVICE=** keyword must precede all other fields. Table 2.6.1 lists the valid device type keywords.

Table 2.6.1: Device Type Keywords

Device Type	Description	Operating Systems
ARI	PASS demo	All
ABI	ABI-PC2 in extended memory configuration.	Windows 3.1/95/98
ADI	ABI-PC2 in DOS configuration	Windows 3.1/95/98
ADI3	ABI-PC3 in DOS large memory window configuration	Windows 3.1/95/98
SADI3	ABI-PC3 in DOS small memory window configuration	Windows 3.1/95/98
SNT3	ABI-PC3 in Windows NT small memory window	Windows NT
SA3C1	Channel 1 of Dual Channel ABI-PC3 (DOS small window)	Windows 3.1/95/98
SA3C2	Channel 2 of Dual Channel ABI-PC3 (DOS small window)	Windows 3.1/95/98
NT3_1	Channel 1 of Dual Channel ABI-PC3 (NT small window)	Windows NT
NT3_2	Channel 2 of Dual Channel ABI-PC3 (NT small window)	Windows NT
SMCIA	ABI PCMCIA (PCM)	Windows 3.1/95/98
SMCIA2	ABI PCMCIA2 (PCM2)	Windows 3.1/95/98
APCI	ABI-PCI	Windows 95/98
NTPCI	ABI-PCI	Windows NT
PC104	ABI-PC104 or ABI-FW5000	Windows 3.1/95/98
A1773	PASS PCI-1773	Windows 95
NT1773	PASS PCI-1773	Windows NT

2.6.3 Base I/O Address Field

PC2 and PC3

The Base I/O Address field specifies the base I/O address in hexadecimal. When using multiple cards, each card must be at a different base address. In addition, each base address must be a multiple of 8h for PC3 cards, or a multiple of 2 for PC2 cards. An example for 2 single channel PC3 cards appears below:

```
DEVICE=SADI3, 390, D0, 15, PC3BRM1, *
DEVICE=SADI3, 390, D0, 15, PC3MON1
DEVICE=SADI3, 3A0, D0, 12, PC3BRM2, *
DEVICE=SADI3, 3A0, D0, 12, PC3MON2
```

For a dual channel board, the Base I/O address of the card (Channel 1) must be a multiple of 8. The Base I/O address of the second channel (Channel 2), must be equal to the address of the Channel 1 plus 6h. An example for a dual channel PC3 appears below:

```
DEVICE=SA3C1, 390, D4, 11, BRMC1, *
DEVICE=SA3C1, 390, D4, 11, MONC1
DEVICE=SA3C2, 396, D4, 11, BRMC2, *
DEVICE=SA3C2, 396, D4, 11, MONC2
```

In both examples above, the base memory address may be the same for all configuration lines for all devices. In the last example above, you could not put another board at base I/O address 398.

PCI, cPCI, and 1773 PCI

Under Windows NT, the Base I/O Address field is always 0.

Under Windows 95/98, this field represents the board number on the PCI bus. The board numbering starts at 0, and multichannel devices cannot have the same board number. For example, two dual channel PCI cards would have 8 lines, the first four lines would have a field value of 0, and the next four lines would have a field value of 1. An analogous situation applies for two single channel devices. the first two lines would have a field value of 0 and the next two lines would have a field value of 1.

Warning: These cards are not supported under Windows 3.1

PCM/PCM2

Under Windows 3.1/95/98, the Base I/O address field is always 0.

Warning: This card is not supported under Windows NT.

2.6.4 Base Memory Address Field

The PASS contains 256 kilobytes of on-board RAM. When the PASS is installed in DOS memory, the host computer views the PASS as either four overlaid 64-kilobyte blocks or sixteen 16-kilobyte blocks at the same address, depending on the selected window configuration. When configured as a large DOS window, the base address of the board can be assigned an address that is on a 64-kilobyte boundary, typically, the 0D 0000h or 0E 0000h segments. When configured as a small DOS window, the base address of the board can be assigned an address that is on a 16-kilobyte boundary, typically, the 0D 0000h or 0D 4000h segments. (Select the desired DOS window configuration by using the appropriate `DEVICE` keyword in the *pass.cfg* file, described under step III-B.)

The DOS memory area consists of the addresses from 00 0000h to 10 0000h. Within this area, certain addresses are reserved by DOS and must not be used by the PASS.

Note: If you 're running Windows 3.x use the DOS **`DEBUG`**, or the **`MSD`** program to read the memory block and determine if it is unused. (If all of the data in the block is FFh, the block is probably unused.) If you're running Windows 95/98, use the System control panel to determine system resource availability.

Table 2.6.2 contains the memory map of the PC/AT and should be used to help determine a possible base address for the PASS.

Table 2.6.2: PC/AT Memory Map

Address Range	Length	Description
000000h - 0BFFFFh	768K Bytes	Reserved for DOS.
0C0000h - 0CFFFFh	64K Bytes	May be used by video adapters.
0D0000h - 0DFFFFh	64K Bytes	Unused or DOS ROM area. This block may be used by certain adaptors or by an Expanded Memory Manager (EMM). This is the recommended region in which to initially map the PASS card.
0E0000h - 0EFFFFh	64K Bytes	Often used by Windows 95/98 systems for plug and play extensions. Use this if the 0D region is unavailable.
0F0000h - 0FFFFFh	64K Bytes	DOS ROM area.
100000h - FFFFFFh	15M Bytes	Extended Memory area. Some or all of this area may be used by extended memory installed in the system.

PC2 and PC3 The Base Memory Address field specifies the most significant eight bits of a 24-bit address in hexadecimal and must be the same for both configuration lines for the device. Base memory addresses are represented in one of the formats shown in Table 2.6.3.

Table 2.6.3: Base Memory Address Formats

Device Type	Base Memory Address	Address Value in *pass.cfg* File
ABI, ADI, ADI3	0D 0000h	0D
SADI3, SA3C1, SA3C2	0D 0000h	D0

For PC2 boards, a base memory address of 0D 0000h is represented as 0D. For PC3 boards memory-mapped in small DOS windows, a base memory address of 0D 0000h is represented as D0 to allow an offset from the base address.

The following examples show the setting of the base memory address for DOS memory space.

A single PC3 board using a small memory model:

```
DEVICE=SADI3, 390, D0, 11, PC3BRM, *
DEVICE=SADI3, 390, D0, 11, PC3MON
```

A single PC3 board using a large memory model:

```
DEVICE=ADI3, 390, 0D, 11, PC3BRM, *
DEVICE=ADI3, 390, 0D, 11, PC3MON
```

Two PC3 boards using a large memory model:

```
DEVICE=ADI3, 390, 0D, 11, PC3BRM1, *
DEVICE=ADI3, 390, 0D, 11, PC3MON1
DEVICE=ADI3, 3A0, 0D, 12, PC3BRM2, *
DEVICE=ADI3, 3A0, 0D, 12, PC3MON2
```

Three PC3 boards using a large memory model:

```
DEVICE=ADI3, 390, 0D, 11, PC3BRM1, M, *
DEVICE=ADI3, 390, 0D, 11, PC3MON1, M
DEVICE=ADI3, 3A0, 0D, 12, PC3BRM2, S, *
DEVICE=ADI3, 3A0, 0D, 12, PC3MON2, S
DEVICE=ADI3, 3B0, 0D, 9, PC3BRM3, S, *
DEVICE=ADI3, 3B0, 0D, 9, PC3MON3, S
```

The last example illustrates the configuration for the external clock option, using one board as a master and the other boards as slaves.

PCI, cPCI, 1773 PCI, PCM, and PCM2

For the PCI, cPCI, 1773 PCI, PCM and the PCM2, the Base Memory Address field is always 0.

2.6.5 Interrupt Level Field

The interrupt level field specifies the interrupt level. The value must be in decimal.

Note: If an interrupt level is selected that is used by another device, unpredictable results will occur.

Table 2.6.4 lists the standard interrupt levels.

Table 2.6.4: Standard Interrupt Levels

IRQ Level:	Description:
15	Second hard drive controller
14	First hard drive controller
13	System reserved
12	PS/2 mouse or free
11	(free)
10	(free)
9	(free)
8	System reserved
7	LPT1
6	Floppy drive
5	(free)
4	COM1
3	COM2
2	System reserved
1	System reserved

PC2 and PC3

Usually, you can use IRQ 3, 4, 5, 9, 10, and 11 for PASS. You can also use IRQ 15 *only if* a second hard drive controller is *not* present.

To verify that no conflict exists, perform the procedure outlined in Section 2.11, Testing Your Installation.

PCI, cPCI, 1773 PCI, PCM, and PCM2

For the PCI, cPCI, 1773 PCI , PCM, and the PCM2, the Interrupt Level field is always 0.

2.6.6 Device Name Field

The device name field contains a unique name for the BC/RT and monitor functions. This field can contain up to 11 characters of your choosing. The entire field appears in the first text box of the status bar.

Warning: For Windows NT devices, the first 5 or 6 characters in the device name field in the first line of each pair (e.g., PC3BRM, PRMC1, BRMC2, PCIBRM) must be as specified as in the sample *pass.cfg* file.

2.6.7 Optional Fields

In the optional field you can specify either the IRIG time source option or the external clock option.

To enable the IRIG decoder circuitry on a card so equipped, place an `I` in the optional field of both lines in the device pair. The IRIG encoder option is only available on ABI-PC3 and ABI-PCI versions.

Note: If you need IRIG capability on your existing ABI-PC3 or ABI-PCI card, contact SBS Technologies for information on upgrading.

When the PASS-1000 supports the External Clock option, the optional field entry **M** is used to indicate which installed PASS device functions as the master, supplying the clock signal. The optional field entry **S** indicates which device functions as a slave, receiving the clock signal. If more than one device is configured as a master, the PASS-1000 software will use the first device listed in the file as the master. All other devices will be configured as slaves.

Cross Reference: See the section *External Connector Pinouts* on page 2-53 for the pinout information pertaining to the External Clock option.

To increase bit-error sensitivity when using a PC3 or PCI card, place a `B` in the optional field of both lines in the device pair.

2.7 Installing the PASS Card into the PC System

This section describes what you need to know in order to physically install the card in your system.

Each version of the PASS card uses a slightly different installation procedure. These versions, the first two digits of their serial numbers, and the corresponding procedures appear in Table 2.7.1.

Warning: If you have a PC2 or a PC3 board, see Section 2.5 for information on finding system resources. You will need this information before physically installing the card.

Table 2.7.1: Installation Procedures

For PASS card version:	Having serial numbers beginning with:	See page:
The boards listed below require jumper modifications.		
PC2	24	2-32
PC3-1	47	2-37
PC3-2	44	2-37
PC104	46	2-41
FW5000	78	2-43
The boards listed below *do not* require jumper modifications. Proceed to the pages given below located in Section 2.8.		
cCPI	–	2-44
PCI-1	57	2-44
PCI-2	75	2-44
PCI-1773	–	2-44
PCMCIA	48	2-49
PCMCIA2	48	2-49

2.7.1 PC2 Cards

Windows 3.x/95/98

The PC2 card requires setting the base I/O address, RAM base address, and Interrupt level via DIP switches on the card prior to installing the card into the PC. For more information, please refer to the PC2 Version of the PASS-1000 manual.

Warning: The PC2 board in not supported under Windows NT.

Setting the Hardware Switches

Three hardware switch settings must be configured for proper operation of the PASS-1000:

- Switch SW3 selects the base memory address
- Switch SW2 selects the interrupt level
- Switch SW1 selects the base I/O address

The following paragraphs provide a detailed discussion of each of these switch settings, as well as information on connecting the PC2 to a 1553 bus and a pinout listing for external connector P4.

Setting the Base Memory Address

The mother board and monitor board of the PC2 board each contain 128 kilobytes of on-board RAM. When the PC2 board is installed in DOS memory, the host computer views the PC2 as four overlaid 64-kilobyte blocks at the same address. The base address of the board can be assigned an address that is on a 64-kilobyte boundary, typically, the 0D 0000h or 0E 0000h segments.

The DOS memory area consists of the addresses from 000000h to 100000h. Within this area, certain addresses are reserved by DOS and must not be used by the PC2. Use the DOS DEBUG program to read the memory block and determine if it is unused. (If all of the data in the block is FFh, the block is probably unused.)

Table 2.7.2 contains the memory map of the PC/AT, and should be used to help determine a possible base address for the PC2.

Table 2.7.2: PC/AT Memory Map

PC/AT Memory Map		
Address Range	**Length**	**Description**
000000h-0BFFFFh	768 KB	Reserved for DOS.
0C0000h-0CFFFFh	64 KB	Unused. May be used by video adapters.
0D0000h-0DFFFFh	64 KB	Unused or DOS ROM area. This block may be used by certain adapters or by an Expanded Memory Manager (EMM). This block can only be a base address if the board is configured in the 64 KB block mode.
0E0000h-0EFFFFh	64 KB	This block can only be a base address if the board is configured in the 64 KB block mode.
0F0000h-0FFFFFh	64 KB	DOS ROM area
100000h-FFFFFFh	15 MB	Extended Memory area. Some of all of this area may be used by extended memory installed in the system.

Switch SW3 is used to select the base address of the PC2. If the desired value of the address line is “0”, the switch should be placed in the ON position; if the desired value is “1”, the switch should be placed in the OFF position. Figure 2.7.1 illustrates the switch positions for base address 0D 0000h, in DOS space. The switch is read from left to right with SW3-8 being the least significant position.

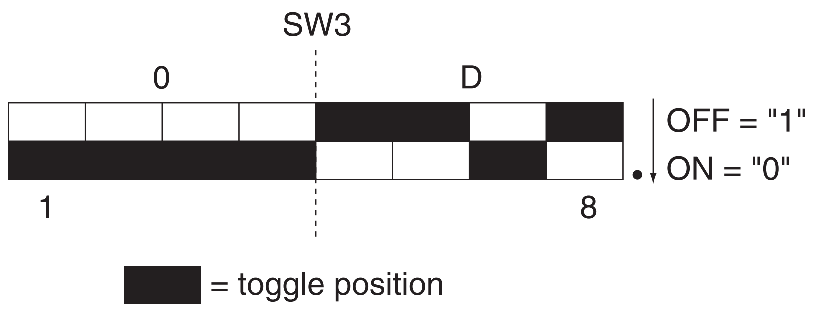


Figure 2.7.1: Sample SW3 Settings

Table 2.7.3 illustrates the proper switch settings for base address 0D 0000h.

Note: Bits 0 through 15 are not selectable and default to zeros.

Table 2.7.3: Example of Base Address of 0D0000h

Example of Base Address 0D0000h			
Address Bit	**Switch**	**Desired Value**	**Switch Setting**
A23	SW3-1	0	On
A22	SW3-2	0	On
A21	SW3-3	0	On
A20	SW3-4	0	On
A19	SW3-5	1	Off
A18	SW3-6	1	Off
A17	SW3-7	0	On
A16	SW3-8	1	Off

Setting the Interrupt Level

Switch SW2 is used to set the interrupt level used by the PC2. You can choose between eight different interrupt levels or no interrupts. The following figure illustrates a PC2 configured for interrupt level 15.

Warning: If your computer has a second hard drive controller, *do not* use IRQ 15.

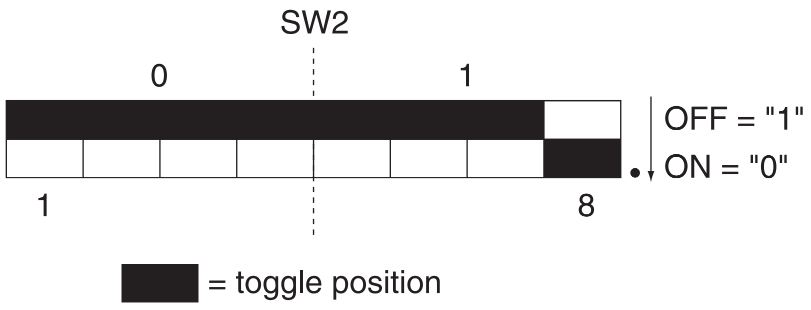


Figure 2.7.2: Sample SW2 Settings

The following table contains a list of the possible interrupt level settings along with their use in an IBM PC/AT system. Only one of the switch positions in SW2 can be ON; if more are ON, undetermined results will occur. Also, if an interrupt level is selected that is used by another device, undetermined results will occur.

Table 2.7.4: Example Interrupt Level Selection

Example of Interrupt Level Selection	
Interrupt Level	**SW2 Switch Settings**
None	All Off
IRQ3	SW2-1 On
IRQ5	SW2-2 On
IRQ7	SW2-3 On
IRQ9	SW2-4 On
IRQ10	SW2-5 On
IRQ11	SW2-6 On
IRQ12	SW2-7 On
IRQ15	SW2-8 Off

Setting the Base I/O Address

Switch SW1 is used to set the base I/O address of the ABI I/O Control Register. The I/O Control Register enables/disables the PC2 and selects the desired page of memory during board accesses. The PASS software automatically controls this register during PASS operation. The user may only be required to write to the I/O Control Register when operating the board as a PC2 card running the full function microcode (i.e., with the interface libraries for a user's custom application). If this is the case, see the ABI Reference Manual for details about the use of the I/O Control Register.

Two consecutive addresses must be available for the PC2: the mother board is accessed at the base I/O address and the monitor board at the next consecutive address. However, the user is only required to set the base I/O address; the PASS software automatically looks at the next consecutive address when accessing the monitor board. The base I/O address of the I/O Control Register can be set to any value between 300h and 3FFh. Valid I/O base addresses (for either extended or DOS memory configuration) are 390h, 392h, and 394h. The eight switches of SW1 correspond to the eight least significant address lines. Figure 2.7.3 and Table 2.7.5 illustrate the correspondence between the switches and address lines, along with an example for the default setting of 390h.

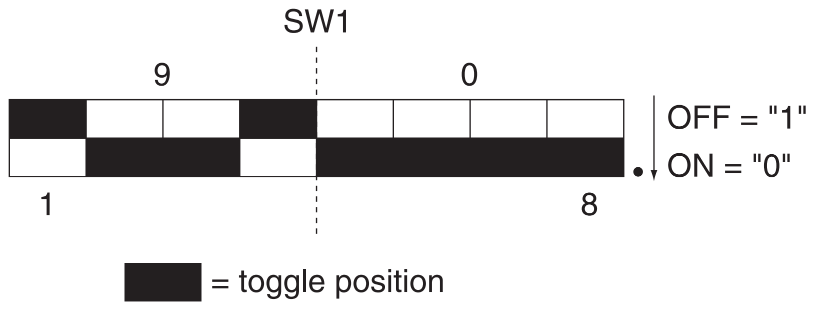


Figure 2.7.3: Sample Setting for SW1

Table 2.7.5: Example of Base Address of 390h

	Example of Base Address 390h		
Address Bit	**Switch**	**Desired Value**	**Switch Setting**
A23	SW3-1	0	On
A22	SW3-2	0	On
A21	SW3-3	0	On
A20	SW3-4	0	On
A19	SW3-5	1	Off
A18	SW3-6	1	Off
A17	SW3-7	0	On
A16	SW3-8	1	Off

2.7.2 PC3 Cards

To install the PC3 version of the PASS-1000 system in DOS memory properly, you need to determine an appropriate base I/O address and set hardware switch SW1 on the board itself. (See Figure 2.7.4 and Figure 2.7.5 for the switch locations on an PC3-1 and an PC3-2 card, respectively.)

Setting the Base I/O Address

To set the Base I/O address of the PC3 I/O Control Register, you use Switch SW1. The I/O Control Register enables/disables the PASS and selects the desired page of memory during board accesses. The PASS software automatically controls this register during PASS operation.

The base I/O address of the I/O Control Register can be set to any value between 000h and 7FFh. Valid Base I/O addresses are 390h, 398h, etc. (the second channel of a dual channel board will be 06h above the first channel). The eight switches of SW1 correspond to address lines 3 through 10, with SW1-1 being the most-significant bit. Figure 2.7.7 and Table 2.7.6 illustrate the correspondence between the switches and address lines, along with an example for the default setting of 390h.

You may only be required to write to the I/O Control Register when operating the board as an ABI-PC3 card running the full function firmware (i.e., the SBS Integrated Avionics Library). If you wish to use this capability, contact SBS to obtain the required software, firmware, and reference manuals at no-cost.

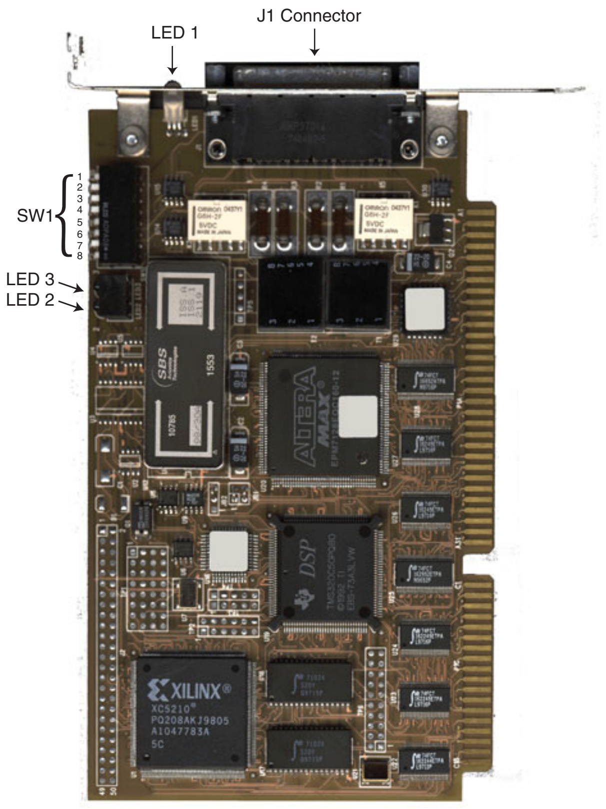


Figure 2.7.4: ABI-PC3-1 Board Layout (ISA 1/2-Size Card)

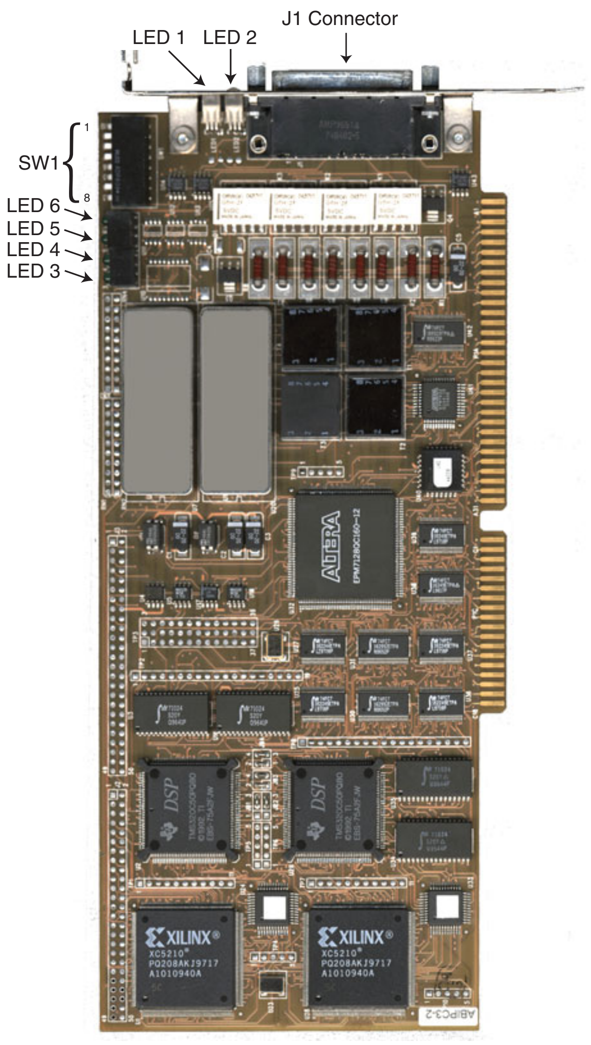


Figure 2.7.5: ABI-PC3-2 Board Layout (ISA 3/4-size card)

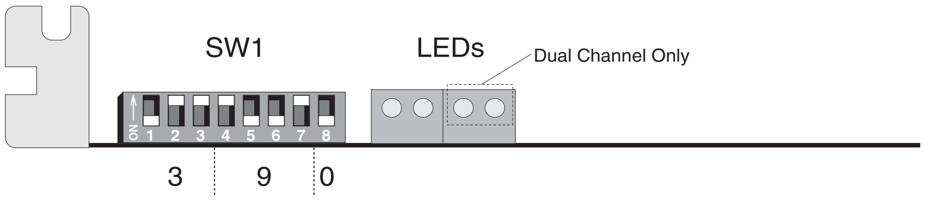


Figure 2.7.6: Edge View of PC3 Board as Installed in Host Computer

Table 2.7.6: SW1 Switch Settings Example

Example of Base I/O Address 390h				
Address Bit	**Switch (SW1)**	**Desired Value**		**Switch Setting**
A10	SW1-1	0	3	OFF
A9	SW1-2	1		ON
A8	SW1-3	1		ON
A7	SW1-4	1	9	ON
A6	SW1-5	0		OFF
A5	SW1-6	0		OFF
A4	SW1-7	1		ON
A3	SW1-8	0	0	OFF

2.7.3 PC104 Cards

To install the PC104 version of the PASS-1000 system in DOS memory properly, you need to determine an appropriate base I/O address and set hardware switch SW1 on the board itself. (See Figure 2.7.7 for switch location)

Setting the Base I/O Address

To set the Base I/O address of the ABI I/O Control Register, you use Switch SW1. The I/O Control Register enables/disables the PASS and selects the desired page of memory during board accesses. The PASS software automatically controls this register during PASS operation.

The base I/O address of the I/O Control Register can be set to any value between 300h and 3FFh. Valid Base I/O addresses are 390h, 398h, etc. The five switches of SW1 correspond to address lines 3 through 7, with SW1-1 being the most-significant bit. Figure 2.7.7 and Table 2.7.7 illustrate the correspondence between the switches and address lines, along with an example for the default setting of 390h.

You may only be required to write to the I/O Control Register when operating the board as an ABI-PC104 card running the full function firmware (i.e., with the SBS Integrated Avionics Library). If you wish to use this capability, contact SBS to obtain the required software, firmware, and reference manuals at no cost.

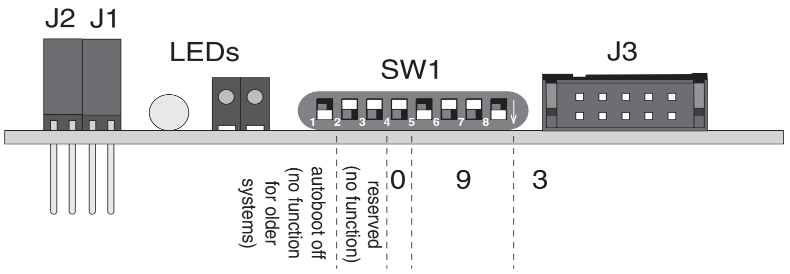


Figure 2.7.7: Edge view of PC104 Board as Installed in Host Computer

Table 2.7.7: SW1 Switch Settings Example (Base I/O Address 390h)

Address Bit	Switch (SW1)	Desired Value		Switch Setting
N/A	SW1-1	Autoboot Off		ON
N/A	SW1-2	Reserved		OFF
N/A	SW1-3	Reserved		OFF
A3	SW1-4	1	0	OFF
A4	SW1-5	0	9	ON
A5	SW1-6	0		OFF
A6	SW1-7	1		OFF
A7	SW1-8	0		ON

2.7.4 FW5000 Cards

To install the FW5000 version of the PASS-1000 system, plug the module into the computer as shown in Figure 2.7.8.

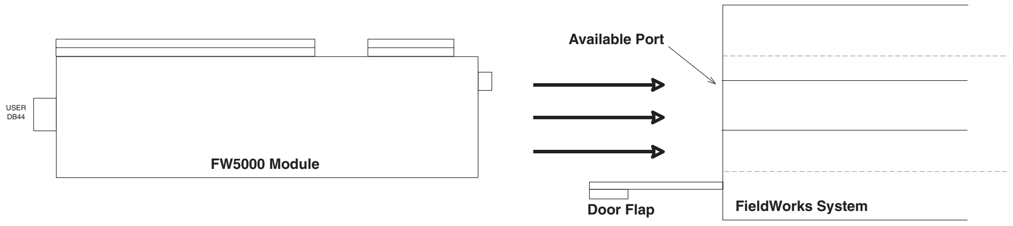


Figure 2.7.8: Inserting the FW5000 Card into the FieldWorks System

The base I/O address can be set to 380h, 388h, 390h or 398h (the default is 380h). On the standard product, add jumpers to the J1 connector to make the appropriate connections listed in Table 2.7.8 (refer to Figure 2.7.9 for the J1 pinouts). Alternately, you can have SBS hard-wire the address within the module.

Table 2.7.8: Base I/O Address Settings

Desired address	Required Jumper on J1 Mating Connector
380h	None
388h	J1-5 to J1-20
390h	J1-6 to J1-21
398h	J1-5 to J1-20 and J1-6 to J1-21

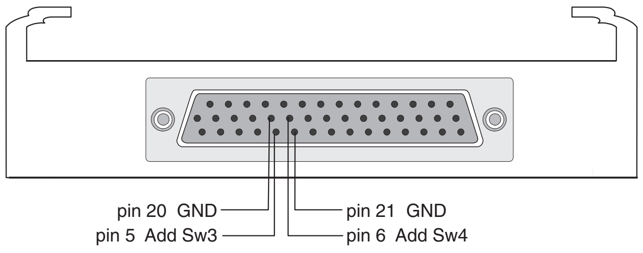


Figure 2.7.9: FW5000 Pins Used for Setting the Base I/O Address

2.8 Installing the OS Specific Device Drivers

Some PASS cards require the installation of operating system specific device drivers. These cards are:

- ➢ cPCI, PCI, and 1773 PCI Cards
- ➢ PCMCIA, PCM, and PCM2 Cards

The procedures for the cPCI, PCI, and 1773 PCI are identical. The procedure for the PCMCIA differs slightly. The remaining subsections describe the driver installation procedure for each card.

2.8.1 cPCI, PCI, and 1773 PCI Cards

To install the drivers for the cPCI, PCI, or 1773 PCI, follow the procedure outlined below for your operating system.

Warning: PASS does not support cPCI and PCI cards installed on the same platform.

Note: The 1773 PCI will not function in 1553A mode.

Windows 95/98

1. After inserting the PASS card into an available PCI slot, power up your PC.
2. Windows 95/98 will display the **New Hardware Found** window. Select **Next**.

   After searching the floppy drives, the Plug and Play Manager prompts you to select another location.
3. Choose **Other Location**.

   A dialog appears asking you to specify the location.
4. Enter **c:\pass\dev_drvr**.

   The installation program examines the contents of *sbspci.inf* file. A prompt appears asking you to select a SBS 1553 PCI Card or a SBS 1773 PCI Card.
5. To select and install the PCI Windows 95/98 driver, click **OK**.

   The PASS 1000 installation is now complete.

Warning: Before using the PASS card, check for a possible memory conflict by completing the steps below.

1. Go to **Settings** » **Control Panel** from the Windows 95/98 Start menu.
2. Double-click on the **System** icon, then click on the **Device Manager** tab.
3. Look at the list of **System devices** (click on the plus [+] sign if one appears next to **System devices**) and locate the SBS Technologies entry.
4. If an exclamation point (!) appears next to the entry, a memory conflict exists. To resolve the conflict, continue on to the next step. Otherwise skip the rest of this procedure and go to the next section to begin the Software installation process.
5. Double-click on the **SBS Technologies** entry to view the **Properties** for the card.
6. Click on the **Resources** tab, then click the **Set Configuration Manually** button.
7. From the manual configuration page, de-select the **Use automatic settings** box.
8. Click on the **Change Setting** button. Select memory ranges that do not conflict with existing devices, then save the settings and restart the machine.

Windows 95/98 Troubleshooting

If after installing your PASS card, your PC does not ask for device driver information, or does not recognize the existence of the PASS card, try using the procedure below to resolve the problem.

1. Click **Start » Settings » Control Panel**.

   The **Control Panel** window opens.
2. Double click the **System** icon.

   The **System Properties** window opens.
3. Click the **Device Manager** tab.

   A listing of all the device drivers installed on your system appears.
4. The name of the device driver varies according to the type of PASS card installed in your system. The table below describes the possibilities. The device driver should have a yellow circle with an explanation mark.

PASS Card:	Driver Name:
cPCI	SBS 1553 PCI
PCI	SBS 1553 PCI
1773 PCI	SBS 1773 PCI
PCM	SBS Technologies
PCM2	SBS Technologies

Note: If your computer has no information about the device whatsoever, the list will contain an entry called **Other**.

5. Double click on the entry.

   A **Properties** window appears.

6. Click on the **Driver** tab.

   Information about the driver appears.

7. Click the **Update Driver** button.

   The **Update Device Driver Wizard** dialog appears with a message asking you if you want Windows to search for the driver. The **Yes (Recommended)** radio button is preselected.

8. Click the **Next** > button.

   Windows searches your computer's hard drive for an updated driver. The search will be unsuccessful and Windows prompts you to continue using the current driver or to search for an updated driver manually.

9. Click the **Other Locations...** button.

   The **Select Other Location** dialog appears.

10. Enter **`c:\pass\dev_drvr`** in the Location field and click **OK**.

    Windows installs the updated driver. If asked to reboot your computer, do so.

Windows NT To install the Windows NT Driver, follow the procedure below.

Note: Repeat this procedure whenever you make a change to the *pass.cfg* file.

Note: The Windows NT Diagnostics application may be used to verify that the driver is running and the proper resources have been allocated to it. If there are any problems, you can use this application to verify that there are no conflicts with the settings reflected in the *pass.cfg* file.

Warning: You *must* edit the *pass.cfg* file before attempting to install the Windows NT driver. You can find the procedure to edit the *pass.cfg* file on 2-18. Failure to modify the *pass.cfg* file correctly will cause the driver installation to fail.

1. Reboot the system, and log in using an account with Administrator privileges.

2. Select **Run** from the **Start** menu (for Windows NT 3.5, select Run from the File Manager).

3. Type `c:\pass\dev_drvr\install.bat` and click **OK**. This should copy the drivers to the *c:\winnt\system32\drivers* directory and update the registry.

4. A message stating *registry has been updated* should be displayed. If there is a message indicating an error, there was a problem with the *pass.cfg* file. Refer to the Edit the PASS Configuration File instructions on page 2-18 to verify that your *pass.cfg* file is correct and then repeat Step 1 through Step 3. If the registry is updated without error, proceed to Step 5.

5. Reboot the system.

6. After the system has rebooted, select **Programs** from the **Start** menu, then **Administrative Tools (Common)**, and then **Windows NT diagnostics**.

7. Click on the **Resources** tab.

8. Click on the Devices button and then double-click on **SBSPCI**.

9. Verify that the board was registered properly.

10. If it did not register properly, determine where the conflict occurred and return to Step 2.

11. If you have a cPCI card, continue with the the rest of this procedure. Otherwise, stop here.

12. Click **Start**, **Programs**, and **Windows NT Explorer**.

13. In the left pane, folder view, select the *c:\pass* folder.

14. Create a sub folder to *c:\pass*, called *save*.

c:\pass\save is the net result.

15. Move *c:\pass\fwpc3.txt* and *c:\pass\fwpc3i.txt* to *c:\pass\save*.

16. Make a second copy of *c:\pass\fwcpci.txt* and rename the file to *c:\pass\fwpc3.txt*

17. Make a second copy of *c:\pass\fwcpcii.txt* and rename the file to *c:\pass\fwpc3i.txt*

The PASS 1000 driver installation is now complete and you can proceed to the next section.

2.8.2 PCMCIA, PCM, and PCM2 Cards

To install the PCMCIA, PCM, or PCM2 version of the PASS-1000 system into your PC, follow the procedure below for your operating system.

Windows 95/98

Warning: Before performing these steps, Microsoft Win 95/98 PCMCIA Socket Services must be installed.

Warning: Before attempting to operate the PCMCIA PASS card, disable any power management functions that may be running on your computer system.

1. Insert the PASS into an available PCMCIA slot.
2. Windows 95 will display the New Hardware Found window. Select **Next**.

   After searching the floppy drives, the Plug and Play Manager prompts you to select another location.
3. Choose **Other Location**.

   A dialog appears asking you to specify the location.
4. Enter **c:\pass\dev_drvr**.

   A prompt appears asking you to select a PASS PC Card.
5. Select **PASS PC Card** and click **OK**.
6. A double "beep" should sound after the driver is installed.

Windows NT

Note: PCMCIA-PASS is not supported under Windows NT.

Windows 3.x

1. Have the card services running on your system. SBS Technologies recommends the use of SystemSoft's CardWizard for Windows 3.x systems.
2. Insert the PCMCIA card into one of the PCMCIA slots in your system.
3. Copy the *sbs_clnt.exe* file from the *\sbs\stdlib\osdepend\pc\windows\sbsdrvr* directory to the *\sbs\stdlib\working* directory.

Note: Other third-party PCMCIA viewer utilities may not recognize the 1553 device, but the device will still work as long as the card services are running (usually indicated by a beep when inserting the card) and the card has not been configured.

Note: Windows 3.x communicates with the card services using the program *sbs_clnt.exe*, which must be located in the working directory at run-time. The client program is launched when the PCMCIA device is opened and automatically configures parameters such as the base I/O address, base memory address, and interrupt request.

Note: SBS does not recommend installation of multiple ABI/ASF-PCMCIA cards in a single host system. Thus, the *sbs_clnt.exe* file does not support multiple PCMCIA devices.

2.9 Miscellaneous Connection Issues

2.9.1 Connecting the PASS to a 1553 Bus

For operation, the PASS must be connected to a 1553 bus. This is not necessary for initial testing of the board. For test purposes, it is sufficient to simply connect standard bus terminators to the A and B bus connectors on the pigtails of the cable adapter.

These MIL-STD-1553 bus connections must be made very carefully. The PASS will not operate properly if the bus connections are not correct.

The MIL-STD-1553 bus consists of a twisted-pair cable with terminators at each end, ranging in length from a few inches to several hundred feet. The physical connection of a subsystem to this bus is called a stub. In MIL-STD-1553 there are two types of stubs defined: (1) long stubs which use transformer coupling and (2) short stubs which use direct coupling. Short-stub connections to a bus are rarely used and are, therefore, not supported by the PASS-1000. Long-stub connections are the most common and require several components. These components include a transformer and two resistors, used at the exact point where the connection is made to the bus. These components are typically enclosed in a small module called a bus coupler. These couplers may provide for connection of only one stub or several stubs. The coupler has two connectors for the bus (usually at opposite ends of the coupler) and a connector for each stub (typically mounted perpendicular to the bus connectors).

Warning: Subsystem stubs will not function properly if connected directly together.

Proper coupling devices must be used for each stub and the bus must be properly terminated. The following figures illustrate the proper and improper connection schemes for long-stub coupling and the terminology used for the various components associated with the bus. A three-stub coupler is shown. In this case, the bus is contained entirely within the coupler. If other devices are to be connected to the bus, one of the terminators must be removed and replaced with a cable connecting this coupler to another coupler. The other coupler could then be terminated or the bus extended further.

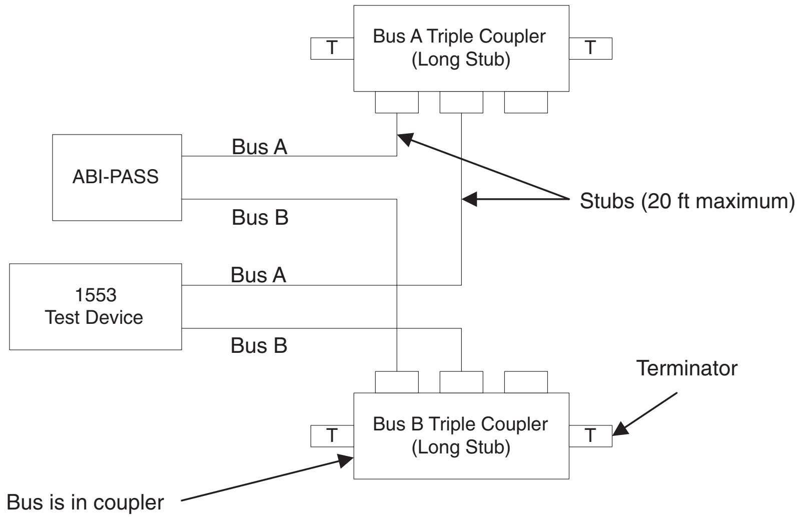


Figure 2.9.1: (For PC3, PCI) Correct Method of Bus Coupling (Long Stubs)

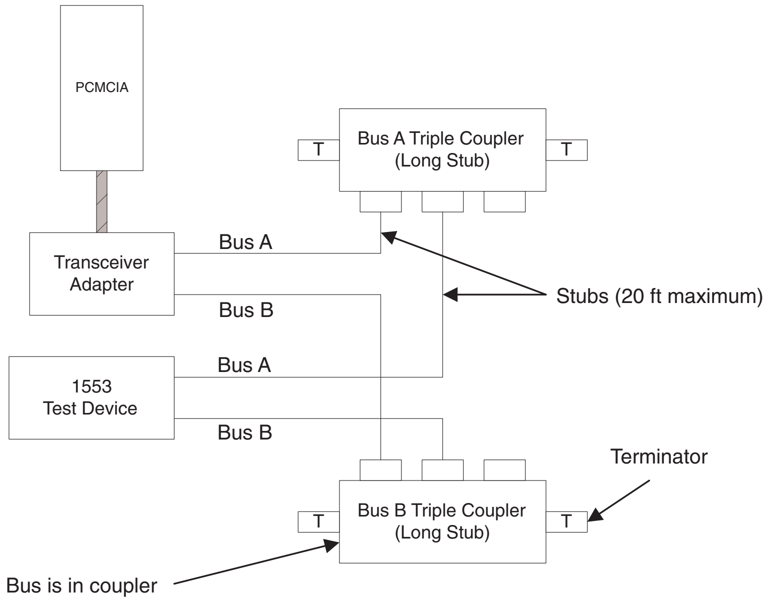


Figure 2.9.2: (For PCMCIA) Correct Method of Bus Coupling (Long Stubs)

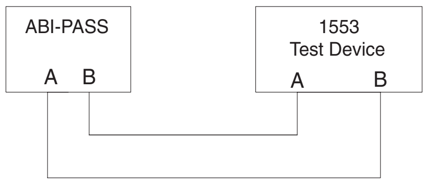


Figure 2.9.3: Incorrect Method of Bus Coupling (Long Stubs)

2.9.2 External Connector Pinouts

Diagrams and pinouts appear on the following pages for these boards:

- ➢ PC3 and PCI
- ➢ PCMCIA
- ➢ PC104
- ➢ FW5000
- ➢ PCI-1773

PC3 and PCI The board is shipped with a cable assembly that attaches to connector J1. A 15-pin connector is attached to one of the leads on this cable assembly. This connector is required for the external trigger, external clock, and IRIG options.

The J1 connector and cable assemblies are shown in Figure 2.9.4 for the PC3-1 and PCI-1 and in Figure 2.9.5 for the PC3-2 and PCI-2. Table 2.9.1 lists the pinout of the 15-pin external options connector.

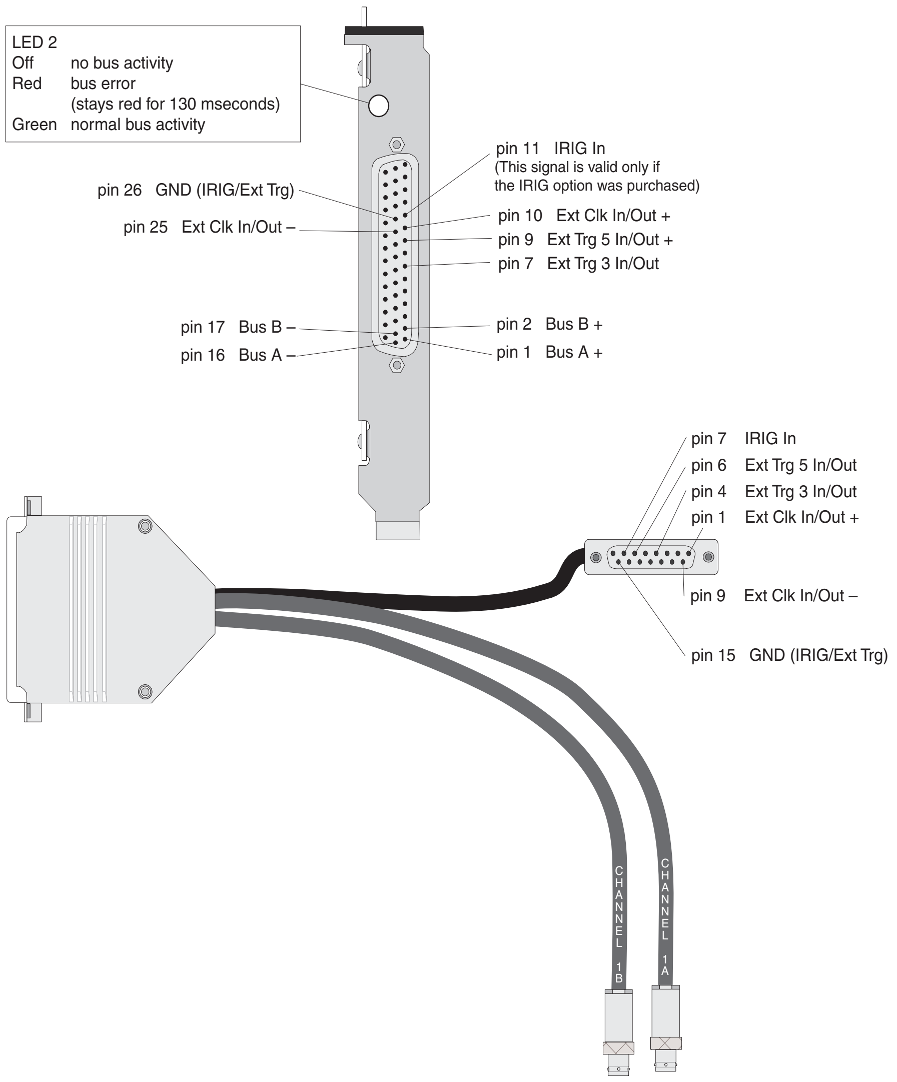


Figure 2.9.4: J1 Connector for PC3-1 and PCI-1

.

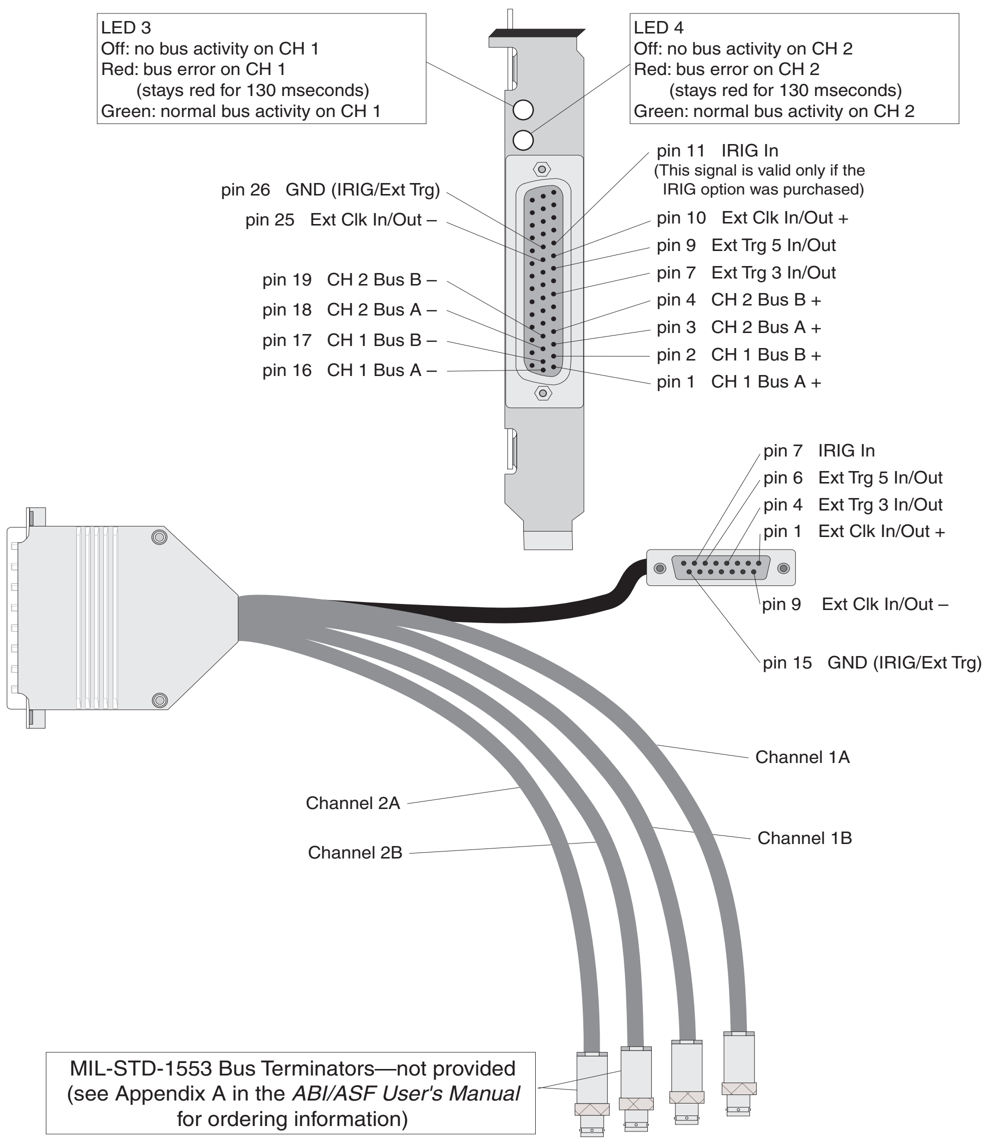


Figure 2.9.5: J1 Connector for PC3-2 and PCI-2

Table 2.9.1: Pinout of External Options Connector for PC3 and PCI Cards

Pin	Corresponding Pin on J1	Signal
1	10	Ext Clock +
2	5	Reserved
3	6	Reserved
4	7	Ext Trg
5	8	Reserved
6	9	Reserved
7	11	IRIG In
8	13	Bias
9	25	Ext Clock –
10	20	Reserved
11	21	Reserved
12	22	Gnd
13	23	Gnd
14	24	Gnd
15	26	Gnd

The Ext Trg signal on pin 4 is active low.

The external clock (master/slave) option may be used to synchronize the time-stamps of multiple PASS devices. To set up this configuration, complete the following steps:

1. In hardware, connect the external clock pins on the 15-pin auxiliary connectors for the devices as follows: connect all Ext Clock + (pin 1) signals together and connect all Ext Clock - (pin 9) signals together.
2. In the *pass.cfg* file, select only one device as the master by placing an `M` at the end of both of the configuration lines for the device. Likewise, designate the remaining devices as slaves by placing an `S` at the end of each of the configuration lines.

When using an external IRIG clock, connect to pin 7.

PCMCIA The card is shipped with a cable assembly that attaches to the PCMCIA transceiver box. The transceiver box contains an RJ-45 connector. Table 2.9.2 lists the pinout information.

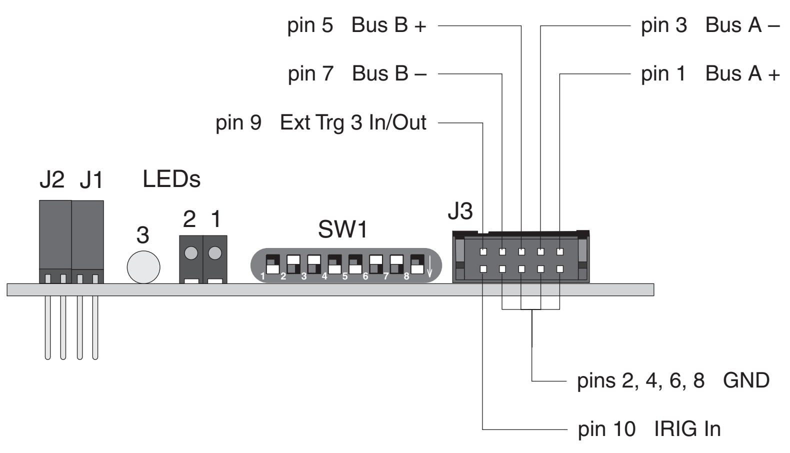
Table 2.9.2: RJ-45 Connector Pinout for PCMCIA Card

Pin	Signal
1	ExtClk+
2	ExtClk–
3	ExtTrg1*
4	ExtTrig2*
5	Bias
6	Gnd
7	Gnd
8	+ 5 VDC

PC104 The board is shipped with a bare connector you can use to fabricate an appropriate cable. Refer to Table 2.9.3 and Figure 2.9.6 for pinout information:

Table 2.9.3: Bare Connector Pinout for PC104 Card

Pin	Signal
1	Bus A+
2	Ground
3	Bus A–
4	Ground
5	Bus B+
6	Ground
7	Bus B–
8	External Trigger
9	Ground



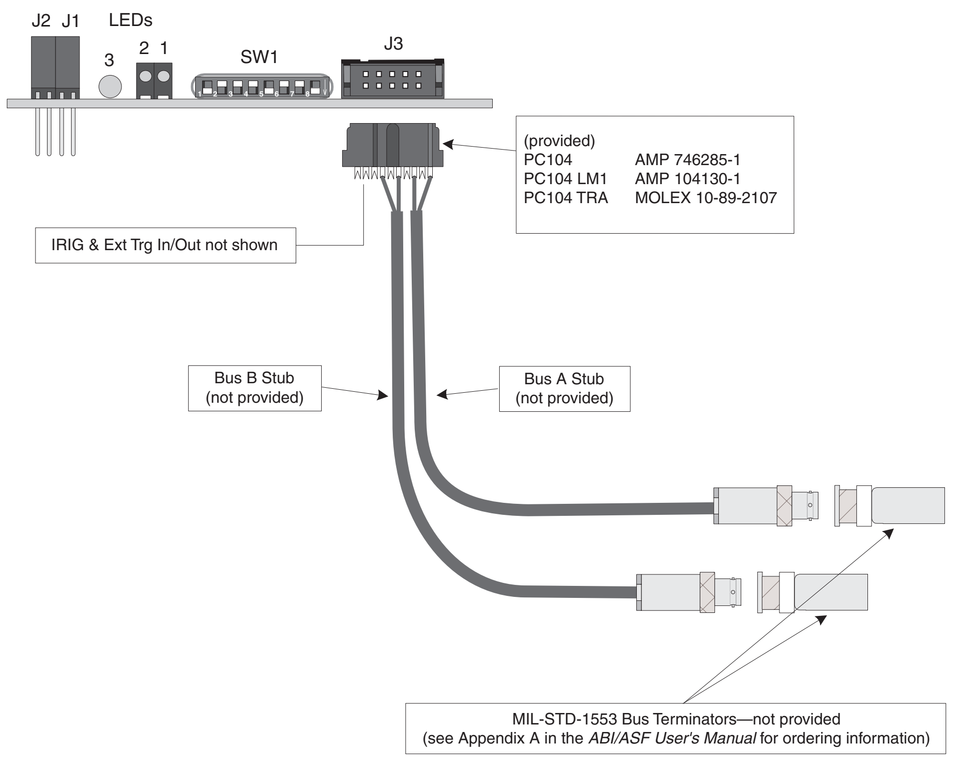

J2 J1
LEDs
3
2 1
SW1
J3
(provided)
PC104 AMP 746285-1
PC104 LM1 AMP 104130-1
PC104 TRA MOLEX 10-89-2107
IRIG & Ext Trg In/Out not shown
Bus B Stub (not provided)
Bus A Stub (not provided)
MIL-STD-1553 Bus Terminators—not provided (see Appendix A in the ABI/ASF User's Manual for ordering information)

Figure 2.9.6: J3 Connector for PC104

FW5000 The board is shipped with a cable assembly that attaches to connector J1. A 15-pin connector is attached to one of the leads on this cable assembly. This connector provides access to Sw3, Sw4, external trigger, and IRIG.

The J1 connector and cable assembly are shown in Figure 2.9.7. Table 2.9.4 lists the pinout of the 15-pin external options connector.

Note: You must provide your own connector if you need to use features from the 44-pin J1 connector that are not accessible from the 15-pin external options connector.

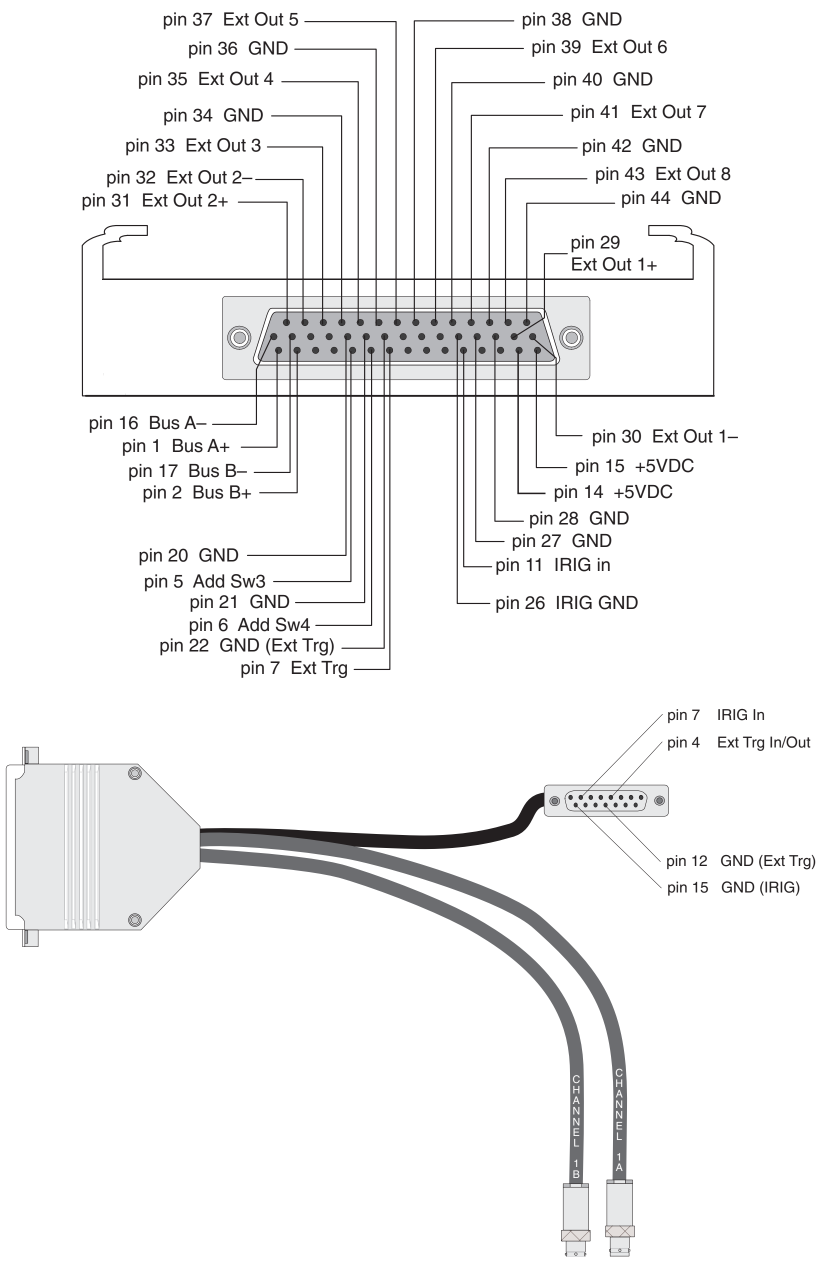


Figure 2.9.7: J1 Connector for FW5000

Table 2.9.4: Pinout of External Options Connector for FW5000 Card

Pin	Corresponding Pin on J1	Signal
1	10	NC
2	5	Add Sw3
3	6	Add Sw4
4	7	Ext Trg
5	8	NC
6	9	NC
7	11	IRIG In
8	13	NC
9	25	NC
10	20	Gnd
11	21	Gnd
12	22	Gnd
13	23	NC
14	24	NC
15	26	Gnd

PCI-1773 An optional Auxiliary Connector Panel can be purchased for the PCI1773. This Panel provides access to auxiliary functions such as external triggers, IRIG input and external clock input. In addition this Panel provides access to a standard 1553 bus when the PCI1773 is configured in 1553 mode. The Auxiliary Panel connectors to the PCI1773 board via connector P2. The Auxiliary Connector Panel is shown in Figure 2.9.8.

Pinouts for the Auxiliary Connector Panel are shown in Table 2.9.5.

Table 2.9.5: Auxiliary Connector Panel Pinouts

PCB Side Header 26-Pin Female	Userside Connector DB44	Signal Name
P1-1	P2-11	IRIG In (Optional)
P1-2	P2-26	Gnd (IRIG)
P1-3	P2-1	1553 Channel A+
P1-4	P2-16	1553 Channel A–
P1-7	P2-2	1533 Channel B+
P1-8	P2-17	1553 Channel B–
P1-11	P2-10	Ext Clock +
P1-12	P2-25	Ext Clock –
P1-14	P2-5	Ext Trg 1+
P1-15	P2-20	Ext Trg 1–
P1-17	P2-6	Ext Trg 2+
P1-18	P2-21	Ext Trg 2–
P1-20	P2-7	Ext Trg 3*
P1-21	P2-22	Gnd (Ext Trg 3)
P1-22	P2-8	Ext Trg 4*
P1-23	P2-23	Gnd (Ext Trg 4)
P1-24	P2-9	Ext Trg 5*
P1-25	P2-24	Gnd (Ext Trg 5)
P1-26	P2-27	Gnd
P1-26	P2-28	Gnd

An ABI/ASF-PCI-1 cable assembly can be utilized to connect to the Auxiliary Connector Panel. The ABI/ASF-PCI-1 cable assembly is shown in Figure 2.9.8. The auxiliary signal pinouts are listed in Table 2.9.5.

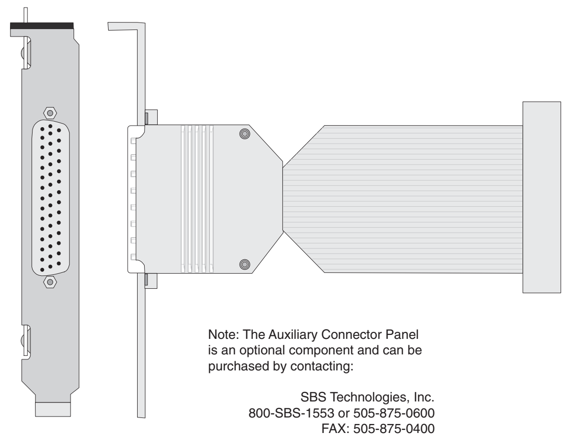

Note: The Auxiliary Connector Panel
is an optional component and can be
purchased by contacting:
SBS Technologies, Inc.
800-SBS-1553 or 505-875-0600
FAX: 505-875-0400

Figure 2.9.8: Auxiliary Connector Panel

Table 2.9.6: ABI/ASF-PCI Pinouts for J1 Connector

Pin	Standard Configuration	IRIG Option
1	Channel 1 A+	Channel 1 A+
2	Channel 1 B+	Channel 1 B+
3	Channel 2 A+	Channel 2 A+
4	Channel 2 B+	Channel 2 B+
5	Ext Trg 1 +	Ext Trg 1 +
6	Ext Trg 2 +	Ext Trg 2 +
7	Ext Trg 3 *	Ext Trg 3 *
8	Ext Trg 4 *	Ext Trg 4 *
9	Ext Trg 5 *	Ext Trg 5 *
10	Ext Clock +	Ext Clock +
11		IRIG Input
12		
13	Bias (1.7VDC)	Bias (1.7VDC)
14	+5VDC	+5VDC
15	+5VDC	+5VDC
16	Channel 1 A–	Channel 1 A–
17	Channel 1 B–	Channel 1 B–
18	Channel 2 A–	Channel 2 A–
19	Channel 2 B–	Channel 2 B–
20	Ext Trg 1–	Ext Trg 1–

Pin	Standard Configuration	IRIG Option
21	Ext Trg 2–	Ext Trg 2–
22	GND (Ext Trg 3)	GND (Ext Trg 3)
23	GND (Ext Trg 4)	GND (Ext Trg 4)
24	GND (Ext Trg 5)	GND (Ext Trg 5)
25	Ext Clock–	Ext Clock–
26	GND	GND (IRIG)
27	GND	GND
28	GND	GND
29		
30-44		

The PCI is shipped with a cable assembly which attaches to the J1 connector. This assembly includes a cable lead with a 15-pin connector for making connections to auxiliary signals. The pinouts for this 15-pin connector are listed in Table 2.9.7.

Table 2.9.7: ABI/ASF-PCI Pinouts for 15-Pin Connector on Cable Assembly

Pin	Corresponding Pin on J1	Signal
1	10	Ext Clock +
2	5	Ext Trg 1+
3	6	Ext Trg 2+
4	7	Ext Trg 3*
5	8	Ext Trg 4*
6	9	Ext Trg 5*
7	11	IRIG In
8	13	Bias
9	25	Ext Clock–
10	20	Ext Trg 1–
11	21	Ext Trg 2–
12	22	Gnd
13	23	Gnd
14	24	Gnd
15	26	Gnd

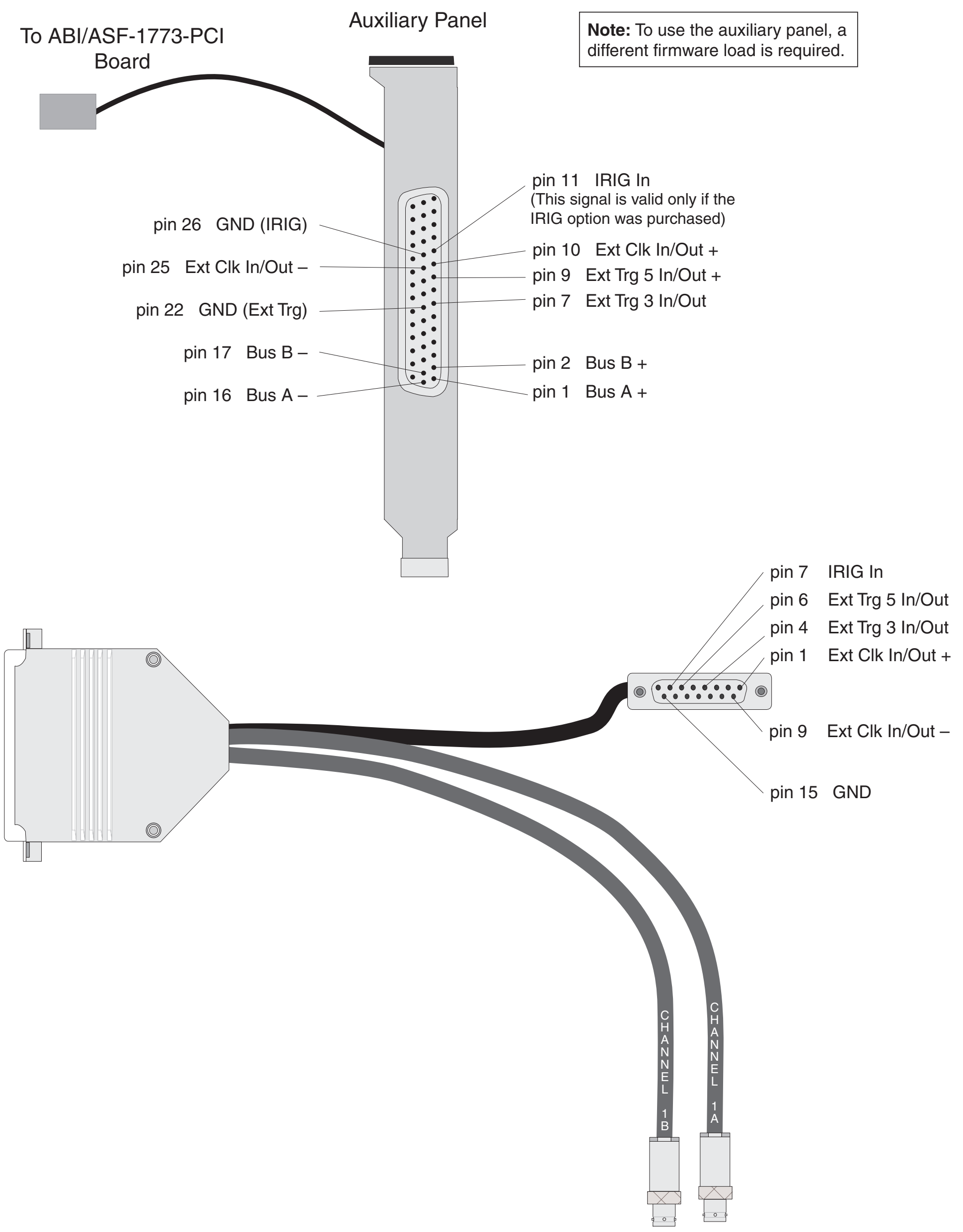


Figure 2.9.9: ABI/ASF-PCI-1 Cable Assembly and Rear Panel Pinouts

2.10 Adding the True-Type Fonts

If you're running Windows 3.x, you need to add the True-Type fonts by following the procedure below.

1. Select **Start** » **Settings** » **Control Panel**. The Control Panel window appears.
2. Double click on **Fonts**. The **Fonts** window appears.
3. Select **File » Install New Font**.... The **Add Fonts** dialog appears.
4. Select **C** as the drive, and **PASS** as the directory.
5. Select **Select All**. to highlight **Lucida Sans Typewriter Bold** and **Lucida Sans Typewriter Regular**.
6. Select **OK**.
7. Select **Close**.
8. Close **Control Panel**. You have completed the software installation process.

2.11 Testing Your Installation

After installing your PASS card and PASS 1000 software, it's a good idea to test that everything is working properly. This section provides you with a test procedure that you can use to test the performance of the various elements in the system.

This procedure is lengthy. However, it provides an extensive test of system I/O, address and interrupts, system performance, and general PASS 1000 functionality.

Warning: Make sure the data cable is connected to the stub before proceeding.

Warning: Make sure *pass.cfg* file has been configured for your PASS card.

Recording Bus Data into an Archive File

1. Start PASS 1000.
2. Click **File** » **Load Current Device Setup File.**

   The **File Open** dialog appears.
3. Select *sbs_test.ba3* (if you're using a PC2, select *sbs_test.bam*) and click **OK**.
4. Click **BC**.

   The **BC Control Panel** appears.
5. To start the data stream, click **Run**.

   A green indicator light for bus activity with an approximate bus loading of 84 percent appears.
6. Click **OK** on the **BC Control Panel**.

   The **BC Control Panel** closes.
7. Select **Monitor** » **Data Logging Mode**.

   The **File Open** dialog appears.
8. Enter a unique filename, then click **OK**.

   The **Monitor Control Panel: Data Logging Mode** window appears.

9. Select the default file size (2179072 bytes) and begin the archive process by clicking the **Run** button.

Note: If a file with the same name exists, you will be prompted to replace the file. Click **Cancel**, then close the **Monitor Control Panel: Data Logging Mode** window and repeat Step 7 and Step 8 being sure to enter a unique filename.

The message **Logging Data...Percent Complete=<X>,** where **X** is a value which increases from **0** to **100** percent. Once the archive process reaches 100 percent, the **Transfer Time** dialog appears.

Note: If the value of **X** has not increased after approximately 5 minutes, your system may not be processing interrupts. Stop here and refer to Subsection 2.11.1 which follows this procedure.

The **Transfer Time** dialog contains information about the archive process. The **File Transfer** field should indicate **Complete**, the **Transfer Time** field should be in the vicinity of 14 seconds (this is system dependent), and the **Number of Bytes Transferred** field should be at least 1986048.

10. If the values you received are different, stop here and refer to Subsection 2.11.1 which follows this procedure. Otherwise, click **OK**.

11. The **Data Logging Results** pop-up window appears. The **Interrupts Generated**, **Interrupts Filled**, and the **Interrupts Written** field should each be at least 36. The last two numbers are significant. They should either be the same value, or differ by 1 from each other (due to PASS 1000 writing data to a partial buffer). If the values you see do not meet these criteria, stop here and refer to Subsection 2.11.1 which follows this procedure.

12. Click **OK**.

    The **View File** dialog appears.

13. To view the results, click **Yes**.

    The **Monitor Control Panel: Data Logging Mode** window reappears with data. If the window appears without data, contact SBS Tech Support for assistance.

Checking for Errors in the Archive

14. The data collected needs to be checked for three types of errors. Click on the **Find** button, located on the right side of the **Monitor Control Panel: Data Logging Mode** window.

    The **Find Specification** dialog appears.

15. Click the **Search from start of buffer** checkbox.

16. Select **Find any error** from the pop up menu at the top of the dialog.

17. To begin the search, click **OK**.

18. The **Search Status** window appears indicating the search progress. At the end of the search, an **Error** dialog appears with the message, **Message not found**. If you received this message, click **OK**. If you *did not* receive this message, then an error was recorded in the archive stream. Stop here and refer to Subsection 2.11.1 which follows this procedure.

19. Click on the **Find** button.

    The **Find Specification** dialog appears.

20. Select **Find msg with Delta Time** <= **Day:Hr:Min:Sec:Msec:Usec** from the pop up menu at the top of the dialog.

21. To begin the search, click **OK**.

    The **Search Status** dialog appears indicating the search progress. At the end of the search, an **Error** dialog appears with the message, **Message not found**. If you received this message, click **OK**. If you *did not* receive this message, then an error was recorded in the archive stream. Stop here and refer to Subsection 2.11.1 which follows this procedure.

22. Click on the **Find** button.

    The **Find Specification** dialog appears.

23. Select **Find msg with Delta Time** >= **Day:Hr:Min:Sec:Msec:Usec** from the pop up menu at the top of the dialog. Enter **050** into the Msec field (the second field from the right).

24. To start the search click **OK**.

    The **Search Status** dialog appears. At the end of the search, an **Error** window appears with the message, **Message not found**. If you received this message, click **OK**. If you *did not* receive this message, then an error was recorded in the archive stream. Stop here and refer to Subsection 2.11.1 which follows this procedure.

25. Click **File** on the **Monitor Control Panel: Data Logging Mode** and then **Exit** to close the window.

    At this point, you have successfully archived a running 1553 data stream with no errors.

Playing Back an Archive File

26. The next step in the testing process is to stop the active stream and perform a playback of the binary archived file to a new binary archive file. Click the **BC** menu.

    The **BC Control Panel** appears.

27. Click **Stop**, then click **OK**.

    The **BC Control Panel** closes and the green stream indicator light turns gray. If the indicator is not gray, please contact SBS Technical Support for assistance.

28. Click the **Playback** menu.

    The **Playback Control** dialog appears.

29. Click **Assign File**.

    The **File Open** dialog appears.

30. To set up your file as the 1553 bus source, select the name you gave the file in Step 8 and click **OK**.

    The **Status** field indicates **Stopped**.

31. Click on the PASS 1000 main window, then select **Monitor » Data Logging Mode**.

    The **File Open** dialog appears.

32. Enter a new output filename different from the name you specified in Step 8 and click **OK**.

    The **Monitor Control Panel: Data Logging Mode** window appears.

33. Click **Run**.

    The message **Waiting for Trigger...** appears.

34. Click on the **Playback Control** dialog to bring it to the front.

35. Click **Run**.

    This starts the playback of the data on the bus, which is, in turn, captured by the new binary archive file you named in Step 32. The **Status** field in the **Playback Control** dialog starts counting the buffers which have been played back on the bus. When playback finishes, the **Status** field indicates **Stopped**.

36. Click on the **Monitor Control Panel: Data Logging Mode** window to bring it to the front.

    The status message should read **Display Logging Data...Percent complete 94%**. The 94% value is normal; playback will always be one buffer less than the original archive. If your results differ, contact SBS Technical Support for assistance.

37. Click **Stop**.

    The **Transfer Time** dialog appears containing information about the archive process. The **File Transfer** field should indicate **Canceled**, the **Transfer Time** field should be approximately **000:00:23.96** (this is platform-dependent), and the **Bytes Transferred** field should be at least 1930880. If the values you received are different, stop here and refer to Subsection 2.11.1 which follows this procedure.

38. Click **OK**.

    The **Data Logging Results** dialog appears. The **Interrupts Generated** field should indicate a value between **36** and **70**, depending on when the playback and archive threads commence, and on how fast your system processes interrupts. The **Interrupts Filled** and **Interrupts Written** fields should each be at least **35** *and* within one count of each other. If they are not, stop here and refer to Subsection 2.11.1 which follows this procedure.

39. Click **OK**.

    The **View File** dialog appears.

40. To view the results, click **Yes**.

    The **Monitor Control Panel: Data Logging Mode** window should now contain data. If not, contact SBS Tech Support for assistance.

Checking for Errors in the Collected Data

41. The data collected needs to be checked for three types of errors. Click **Find**.

    The **Find Specification** dialog appears.

42. Click the **Search from start of buffer** checkbox.

43. Select **Find any error** from the pop up menu at the top of the dialog.

44. To begin the search, click **OK**.

    The **Search Status** window appears indicating the search progress. At the end of the search, an **Error** dialog appears with the message, **Message not found**. If you received this message, click **OK**. If you *did not* receive this message, then an error was recorded in the archive stream. Stop here and refer to Subsection 2.11.1 which follows this procedure.

45. Select **Find msg with Delta Time <= Day:Hr:Min:Sec:Msec:Usec** from the pop up menu at the top of the dialog and make sure all fields are set to zero.

46. To begin the search, click **OK**.

    The **Search Status** window appears indicating the search progress. At the end of the search, an **Error** dialog appears with the message, **Message not found**. If you received this message, click **OK**. If you *did not* receive this message, then an error was recorded in the archive stream. Stop here and refer to Subsection 2.11.1 which follows this procedure.

47. Select **Find msg with Delta Time >= Day:Hr:Min:Sec:Msec:Usec** from the pop up menu at the top of the dialog and enter **050** into the Msec field (the second field from the right).

48. To begin the search, click **OK**.

    The **Search Status** window appears indicating the search progress. At the end of the search, an **Error** dialog appears with the message, **Message not found**. If you received this message, click **OK**. If you *did not* receive this message, then an error was recorded in the archive stream. Stop here and refer to Subsection 2.11.1 which follows this procedure.

49. On the menu bar of the **Monitor Control Panel: Data Logging Mode** window, select **File » Exit**.

The **Monitor Control Panel: Data Logging Mode** window closes.

50. To close the Playback dialog, click **OK**.

You have successfully played back a binary archive file to another binary archive file with no errors. This is the best way to exercise the I/O, Interrupt and memory resources for the PASS card.

2.11.1 If You Encounter an Error

If you encountered an error in the test procedure, it could be due to a:

- ➢ Windows NT installation with a shared memory range or card address
- ➢ Windows 3.1/95/98 shared IRQ value
- ➢ Windows 3.1/95/98 shared address value
- ➢ Windows 3.1/95/98 shared memory range
- ➢ Incorrectly installed device driver
- ➢ Faulty hardware component

To isolate the problem, follow the procedure below.

1. Start PASS 1000
2. Click **Tools** » **Device Status**.

   The **Device Status** dialog appears.
3. Check to see that all devices listed have a status of **OK**. If not, ensure the resources allocated to each device are correct.

Cross Reference: For more information about the pass.cfg file, see Editing the PASS Configuration File on page 2-18. For more information about allocating system resources, see Allocating System Resources on page 2-8.

4. If all devices have an "OK" status, quit PASS 1000.
5. Check the hard drive by running the Scandisk program. Table 2.11.1, "Scandisk Procedures," on page 73 describes the procedure for running Scandisk from each operating system.

Table 2.11.1: Scandisk Procedures

If you are running:	Then:
Windows 3.1	Exit Windows and at the DOS prompt, enter `Scandisk`.
Windows 95/98	Close all programs. Select **Start** » **Programs** » **Accessories** » **System Tools** » **Scandisk**.
Windows NT	Close all programs. Double click on the **My Computer** icon. Right click on the the hard drive used to boot the computer and select **Properties** from the pop up menu. Click the **Tools** tab. Click **Check Now.** Click the **Automatically fix file system errors** and the **Scan for and attempt recovery of bad sectors** checkboxes in the dialog. Click **Start**.

6. Use the File Manager or the Windows Explorer program to ensure there is a minimum of 10MB of available disk space.
7. Verify that Channel A and Channel B are not connected to the same Bus (stub).
8. Verify that Channel A and Channel B are properly terminated.
9. After completing this procedure, try repeating the test procedure beginning on 2-66. If the problem still persists, please contact SBS Technical Support.

2.12 Uninstalling the PASS-1000 Software

To uninstall the PASS software from your system, follow the appropriate procedure below.

Windows 95 & 3.x

1. If you have the installation or upgrade disk for that version, run the Uninstall program. Otherwise, save the *pass.cfg* file and any setup files you wish to keep in the *c:\pass\dbase* directory, then delete (or rename) the *c:\pass* directory.
2. Delete any remaining files in the *c:\pass* directory.
3. Delete any environment variable definitions for the PASS and PASSDB from the *autoexec.bat* file, (e.g., SET pass=c:\Pass and SET passdb=c:\pass\Passdb).
4. Reboot the system to restart Windows 95.

Windows NT

1. If you have the installation or upgrade disk for that version, run the Uninstall program. Otherwise, save the *pass.cfg* file and any setup files you wish to keep in the *c:\pass\dbase* directory, then delete (or rename) the *c:\pass* directory.
2. Delete any remaining files in the *c:\pass* directory.
3. Select **Start**, select **Settings**, select **Control Panel**, and open the folder for System.
4. Select the Environment tab.
5. In the User Variables windows, delete the environment variables for PASS and PASSDB.
6. Reboot the system to restart Windows NT.

2.13 Disabling Adobe Type Manager

If Adobe Type Manager is installed on the PC system, it must be disabled in order for the PASS-1000 software to operate properly. To disable it, complete the following steps:

1. From the Windows *Program Manager*, open the **Main** group window and choose the **ATM Control Panel** icon.
2. Choose **Off** from the ATM box and click **Exit**.
3. Restart Windows.

3: User's Guide

The Protocol Analysis and Simulation System (PASS-1000) performs simultaneous, independent simulation of full bus controller (BC), 31 remote terminals (RT), and bus monitor. The system operates on an IBM compatible personal computer (PC) and is based upon the SBS Technologies Advanced Bus Interface (ABI) PC or PCMCIA board and PASS-1000 window software running under Microsoft Windows 3.x. The system provides an intuitive user interface and advanced features such as data logging to disk, full reconstructive playback, bus loading percentages, activity displays, and error injection/detection.

This chapter explains how to use PASS-1000 to initialize RTs, define and execute a BC list, and monitor and view data. The operations of the PASS-1000 are described within the following sections:

Section Name	Description	Beginning on Page:
PASS-1000 Basics	Introduces the PASS-1000 window and explains how to use the help facilities.	3-2
Bus Controller Simulation	Explains how to set up a bus list and execute it	3-14
Remote Terminal Simulation	Explains how to emulate one or more RTs and to view subaddress data buffers.	3-30
Bus Monitoring	Explains how to capture and view bus traffic.	3-41
Data Logging	Describes how to log streams of data to a device such as a disk or tape.	3-63
Playback	Describes how to reproduce the bus traffic recorded in a PASS-1000 archive file.	3-74
Utilities	Describes several tools that may be used to view bus activity, inspect memory, display status, and adjust the output level.	3-76

This manual assumes the reader has previous knowledge of MIL-STD-1553 operations. If unfamiliar with MIL-STD-1553, please refer to the document An Interpretation of MIL-STD-1553 from SBS Technologies, Inc., included with the PASS shipment.

3.1 Definition of Terms

There are a few terms used frequently throughout this guide that should be understood before proceeding. They are defined below.

Device denotes an ABI-PASS connected to a dual-redundant, MIL-STD-1553 bus.

ABI-PASS refers to any PASS hardware board, such as the ISA PC2, ISA PC3, PCI, PCMCIA, cPCI, etc.

Spurious data refers to unexpected data appearing on the bus.

Word characteristics are used by MIL-STD-1553 to define the format, structure, and encoding of command, status, and data words. Injecting an error involves changing a characteristic of a word.

3.2 PASS-1000 Basics

Before using the PASS-1000, have a firm understanding of how to use Microsoft Windows. The PASS-1000 window software has a graphical interface which uses standard Windows conventions for selecting icons, menus, menu items, and options. The PASS-1000 requires a mouse. The mouse is used to choose and select menus and menu commands, use the Control menu, work with dialog boxes, choose options, and use the help menus.

Cross Reference: For more information, refer to the *Microsoft Windows User's Guide*.

3.2.1 Entering Text

When required, the PASS-1000 will request that additional information be typed into a Text Box. Windows procedures are used for entering, deleting, and editing text. When text is to be typed, an insertion point (flashing vertical bar) marks where the text will be entered. Some windows will have the insertion point already positioned in a text box, but most text boxes must be selected before text can be entered or changed. To place the insertion point within a text box, place the mouse cursor over the text box and click or double-click the left mouse button. Table 3.2.1 describes key combinations that can be used to select and edit PASS-1000 text boxes.

Table 3.2.1: Keystrokes for Selecting and Editing PASS-1000 Text Boxes

If you want to:	Then:
Delete the character or selected text to the left of the insertion point.	Press BACKSPACE
Delete the character or selected text to the right of the insertion point.	Press DELETE
Place the insertion point within a text box.	Click the LEFT MOUSE BUTTON
Select all the text within a text box	Double click the LEFT MOUSE BUTTON
Move the insertion point to the left by one character.	Press LEFT ARROW
Move the insertion point to the right by one character.	Press RIGHT ARROW
Move the insertion point to the next text box.	Press RETURN
Select the text in the next text box.	Press TAB

Cross Reference: More information can be found in *Working with Text* in Chapter 2 of the *Microsoft Windows User's Guide*.

3.2.2 Starting the PASS-1000

To start the PASS-1000 application, follow the procedure below.

1. From Windows, go to the Program Manager window.
2. Open the SBS Applications group window if it is not already open.
3. From the group of icons, choose the PASS-1000 icon by double-clicking the icon, or use the arrow keys to move the selection cursor to the icon and then press ENTER. The PASS-1000 window shown in Figure 3.2.1 will appear.

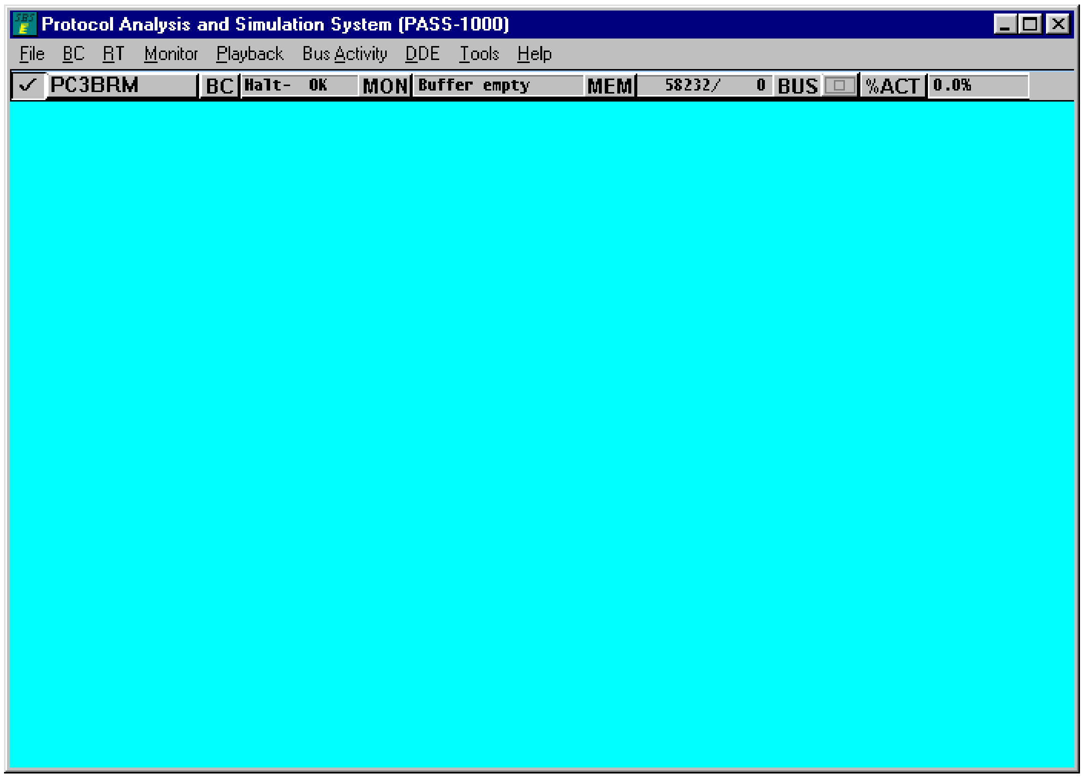

Protocol Analysis and Simulation System (PASS-1000)
File BC RT Monitor Playback Bus Activity DDE Tools Help
PC3BRM BC Halt- OK MON Buffer empty MEM 58232/ 0 BUS %ACT 0.0%

Figure 3.2.1: PASS-1000 Window

If the system is configured incorrectly, one or more error messages may be displayed and the main menu functions will be inaccessible. Be sure to connect the cable assembly to the front of the ABI-PASS.

Cross Reference: See the *Installation* chapter of this manual for further details.

3.2.3 Starting the PASS-1000 Using the *passopen.ini* File

The passopen.ini file allows a previously saved PASS configuration to be loaded and run automatically when the software is started. Using *passopen.ini*, for example, a BC list may be loaded and run automatically.

When the PASS software is started, it checks for a valid *passopen.ini* file. If one exists, the PASS is opened and initialized with values from the file. If a passopen.ini is not found, the PASS software opens to the main window as shown in Figure 3.2.1.

This file contains the following three sections:

[LOAD SETUP]	This section loads a setup file for each of the specified devices. It must contain a separate line for each device to be loaded, in the following format: `Device#=path of setup file to be loaded`
[START BC]	This section starts the transmit function for each device named, *if that device has a transmit list loaded via the LOAD SETUP section*. The format is **`DEVICE=N`**, where **`N`** is the device number from 1 to 8.

[START MONITOR]	This section starts a data logging with the specified file. This section is independent of the first two sections. The size field is optional. If the size is omitted, a default size of 2,200,000 bytes is used.
	
	`File=path of data logging file (required item)`
	`Size=size of data logging file (optional item)`

Warning: The *passopen.ini* file offers very little recovery. Use with caution.

3.2.4 PASS-1000 Window

The PASS-1000 window is composed of the following items:

Menu Bar The Menu Bar lists the menus available on the PASS-1000.

Status Bar The Status Bar displays information about the current simulation. There is one button to select the device and one to clear the bus light indicator.

Title Bar The Title Bar displays the name of the application: SBS Technologies 1553 Bus Analyzer. Click and drag the bar to reduce the size of the window. Title bars for pop-up windows will sometimes display the name, specified in the *pass.cfg* file, that is associated with the device.

Control-Menu Box The Control-menu box in the upper left corner of the window is used to resize, move, maximize, minimize, and close the window and switch to other applications.

To reduce the window to an icon, click the **Minimize** button located in the upper right corner.

3.2.5 Menu Bar

The Menu Bar lies under the Title Bar and displays eight menus that offer a list of commands or actions which may be executed to setup and run a simulation. These menus are described in Table 3.2.2.

Table 3.2.2: Menu Bar Options

Menu	Use it to:
File	Load and save the state of the PASS-1000 from or to a file. Exit the program from this menu.
BC	Setup and execute a bus list
RT	Emulate one or more RTs and to view subaddress data buffers.
Monitor	Capture data. Display the data in a quick snapshot or log it to disk in an archive file.
Playback	Reproduce bus traffic recorded in a PASS-1000 archive file.
Bus Activity	Quickly view bus activity occurring on each RT and their subaddresses
Tools	Inspect memory, view status, adjust the voltage output level, configure the ABI-PASS internal clock, choose 1553A or 1553B protocols for each RT, assign labels for each RT and define 1553A mode codes.
Help	Get on-screen help with every PASS-1000 function.

3.2.6 Using Help

Selecting Help from the Menu Bar will display the PASS-1000's own help system beginning at the Table of Contents page. From this page, more information on any topic can be obtained by clicking on the desired word or phrase. Select Search to search for help on a topic by keyword; select the browse buttons << and >>, when they are lit, to browse through a topic.

3.2.7 Status Bar

Directly under the Menu Bar is the Status Bar that supplies information about the activity occurring within the PASS-1000. This information includes the device name, BC status, Monitor status, the amount of available memory, and bus activity, and the percentage of bus utilization.

The PASS-1000 supports from one to eight devices. (The term device is used to denote an ABI-PASS connected to a dual-redundant, MIL-STD-1553 bus.) A Status Bar appears for each device that is supported and defined in the configuration file named *pass.cfg*.

Cross Reference: See the "Installation" section of this manual for more details about the *pass.cfg* file.

3.2.8 Device Name

The first text box of the Status Bar displays the name for the ABI-PASS board found in the *pass.cfg* configuration file, and it indicates whether or not the device is selected. A check box appears on the left side of the device name. It is used to select (denoted with a check mark) a device from multiple devices. All menu operations except monitoring are performed with respect to the selected device. If there is only one device in the PASS-1000, this check box will always be selected. Windows that are open when a new device is selected remain associated with the device from which they were opened.

3.2.9 BC Status

The next text box displays the activity of the bus controller function. The three possible states are shown in Table 3.2.3.

Table 3.2.3: BC Function States

This state:	Means that the BC:
Halted-OK	is halted and no errors occurred on the last message executed.
Halted-error	has halted due to an error. The corresponding error code is displayed. Error codes are found in Appendix A.
Run-count	is running. A counter is displayed specifying the number of times the BC list has executed. It is incremented continuously.

Monitor Status This text box displays the activity of the bus monitor function. The four possible states are shown in Table 3.2.4.

Table 3.2.4: Bus Monitor Function States

This state:	Means that:
Buffer empty	No data has been stored.
Done	Storager of bus data has been completed.
Storing . . #	Bus data is being stored into the monitor buffer. The pound sign (#) represents the number of unused words remaining in the monitor buffers. During data logging, this number will continuously cycle.
Waiting	The bus monitor is waiting for a trigger condition to occur before storing data.

Memory Status This text box displays the amount of memory available on the ABI-PASS board. Two pieces of information separated by a slash are displayed. The value to the left of the slash shows the amount of unfragmented memory that is available, and the value to the right indicates the amount of fragmented memory that is available. Memory will be compressed to remove fragmented memory whenever a BC bus list is executed.

Bus Activity The next region of the Status Bar is the bus activity button. The color of the button will change depending upon the present and past state of the bus. The color of the inner box indicates past conditions; the color of the outer box indicates present conditions.

This activity button:	Means that:
⬜ (Gray)	There is no bus activity.
⬜🟥 (Gray with a red box)	The bus is currently quiet but an error has been detected in the past.
🟥 (Red)	The bus is active and errors are present.
🟨 (Yellow)	The bus is active and only “no response” errors are present.
🟩🟨 (Green with a yellow box)	The bus is active with no errors but a “no response” error has been detected since the program began monitoring the bus or since the bus activity button was last cleared.
🟩🟥 (Green with a red box)	The bus is active with no errors but an error has been detected since the program began monitoring the bus or since the bus activity button was last cleared.
🟩 (Green)	The bus is active and no errors are present.

Past bus errors may be cleared by clicking the button.

Percentage of Bus Activity

The last text box displays the percentage of bus activity. Bus activity is calculated by dividing number of words appearing on the bus in a given time period by the maximum number of words that is possible. (SBS Technologies calculated the maximum number of words/second to be about 49,200. This value is based upon BC-RT or RT-BC messages with 32 data words and minimum intermessage and status response times.) The percentage is calculated and displayed approximately every half second.

3.2.10 Loading Files

The Load Current Device Setup File option in the File menu will read a PASS binary save file for the currently selected device (denoted by a check mark [✓] in the Status Bar) and reinitialize the PASS-1000 with the memory setup stored in the file. By default, all memory save files for the PC3 and PCMCIA versions of the ABI-PASS have the extension *.BA3*. To load a file, choose the source directory from the Directories list and choose a file from the Files list. If the file was saved with an extension other than the default, place the cursor in the Filename text box and click or double-click the left mouse button. Type the filename and click OK.

A test file called *SBS_TEST.BA3* is included with the PASS software. It includes various BC to RT and RT to BC configurations and may be used to demonstrate different functions of the PASS-1000. To use, install bus terminators on the ABI-PASS board and then select *SBS_TEST.BA3* from the Files list described above. Finally, select BC and choose the Run option. While *SBS_TEST.BA3* is running, the bus activity button on the Status Bar should be green and the BC Status box should display Run with an incrementing counter. If these conditions are not present, contact SBS Technologies.

Note: Memory save files for the PC3 and PCMCIA versions of the ABI-PASS have the extension *.BA3*; memory save files for the PC2 version of the ABI-PASS have the extension *.BAM*. PC2 save files may be converted to the PC3/PCMCIA format and vice versa using the File Conversion option in the File menu, described below.

3.2.11 Saving Files

The Save Current Device Setup as File option in the File menu will save the entire state of the currently selected device (denoted by a check mark [U] in the Status Bar) to a binary file. This option saves all 128K words of ABI-PASS memory and internal variables so that a simulation can be loaded and recreated in the future using the Load Current Device Setup File option. The current BC bus list, RT setup, and monitor data are all saved for the device that is selected.

The file is stored in a binary format and will have a default filename extension of .BA3 for an PC3 or PCMCIA version of the ABI-PASS. If the BC is running, it must be stopped before a save can be performed. In the event of a difficulty encountered with the PASS-1000, the resulting file may be returned to SBS Technologies for analysis.

To save a simulation setup, select the destination directory from the Directories list. Place the cursor over the Filename text box and click or double-click the left mouse button. Enter the desired filename. Alternatively, the file may be saved to an existing file that is displayed in the Files list. Place the cursor on the line containing the filename and double-click the left mouse button. If an existing filename is used, the contents of that file will be lost and replaced with new data. Click OK to save.

3.2.12 Load ASCII File

The Load ASCII File option in the File menu will read a BC, RT, or Misc. ASCII save file and load the PASS-1000 with the data stored in the file. To load an ASCII file, select Load BC List, Load RT Setup, or Load Misc. Setup and choose a file from the Files list. An ASCII file may be created from a PASS setup using the Save as ASCII File option (see below).

3.2.13 Save as ASCII

The Save as ASCII File option in the File menu will save the BC List, RT Setup, or Misc. Setup of the PASS-1000 to an ASCII file. To save as an ASCII file, select Save BC List, Save RT Setup, or Save Misc. Setup. Enter the desired filename and click OK to save. The ASCII file may be edited using any spreadsheet software and most word processing software.

Cross Reference: See Appendix C for details on editing PASS-1000 ASCII files.

Save ASCII Data in Decimal Select this option in the File menu prior to selecting Save as ASCII to create an ASCII file containing decimal data words. A check mark () appears next to this option when it is selected. This is the default setting.

Save ASCII Data in Hex Select this option in the File menu prior to selecting Save as ASCII to create an ASCII file containing hexadecimal data words. A check mark () appears next to this option when it is selected.

3.2.14 Load Default Setup File - Single Device Configured

The Load Default Setup File option only appears in the File menu when a single device is configured in the PASS. It loads the default setup file (*device#.dff*) for the device. File *device.dff* is loaded when this option is selected.

3.2.15 Load Default Files - Multiple Devices Configured

The Load Default Files option only appears in the File menu when two or more devices are configured in the PASS. It loads the default setup file (*device#.dff*) for one or more of these devices. Select a device from the submenu list to load the setup file for a single device. Select Load Default Setup Files for All Devices to load the setup files for all the configured devices. For example, if two devices are configured, files *device1.dff* and *device2.dff* will be loaded when the Load Default Setup Files for All Devices option is selected.

3.2.16 Save Current Setup as Default File - Single Device Configured

The Save Current Setup as Default File option only appears in the File menu when a single device is configured in the PASS. It saves the current settings for the device as the default setup for that device. The default file name is *device1.dff*.

3.2.17 Save Default Files - Multiple Devices Configured

The Save Default Files option only appears in the File menu when two or more devices are configured in the PASS. It saves the current settings of one or more of the devices as the default setup for those devices. The default file name is *device#.dff*, '#' being the device number. Select a device from the submenu list to save the current settings for a single device. Select Save All Current Setups as Default Setup to save the current settings for all the configured devices. For example, if two devices are configured, the current settings will be saved in files *device1.dff* and *device2.dff* when the Save All Current Setups as Default Setup option is selected.

3.2.18 File Conversions

Select the File Conversions option in the File menu to convert a PASS-1000 save file to another file format.

.

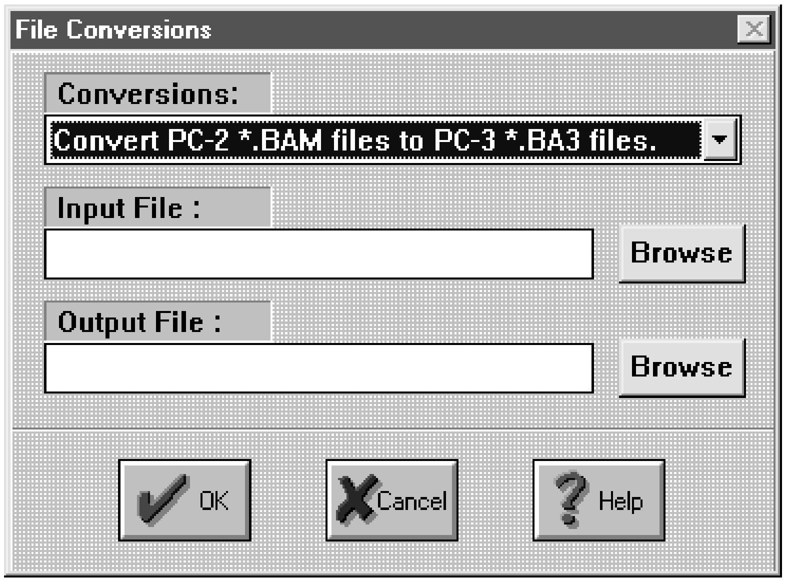


Figure 3.2.2: File Conversions Dialog

The File Conversions dialog has three parameters that must be specified:

Conversions Select one of the following conversion types from the list in the drop-down box:

- ➢ Convert PC-2 *.BAM files to PC-3 *.BA3 files.
- ➢ Convert PC-3 *.BA3 files to PC-2 *.BAM files.
- ➢ Convert Binary archive file to ASCII text.
- ➢ Convert ASCII text archive file to Binary.

Input File Specify the path and filename of the file which is to be converted (the source file). Click the Browse button to open the File Open dialog. Choose the source directory from the Directories list and choose a file from the Files list.

Output File Specify the path and filename of the converted file (the destination file). Click the Browse button to open the File Open dialog. Select the destination directory from the Directories list. Place the cursor over the Filename text box and click or double-click the left mouse button. Enter the desired filename. Alternatively, the file may be saved to an existing file that is displayed in the Files list. Place the cursor on the line containing the filename and double-click the left mouse button. If an existing filename is used, the contents of that file will be lost and replaced with new data.

Specify the Conversion type, the Input File, and the Output File and click OK to effect the conversion.

3.2.19 Quitting

Quit the PASS-1000 by using the Exit command in the File menu. Prior to exiting, save the setup so the simulation may be recreated.

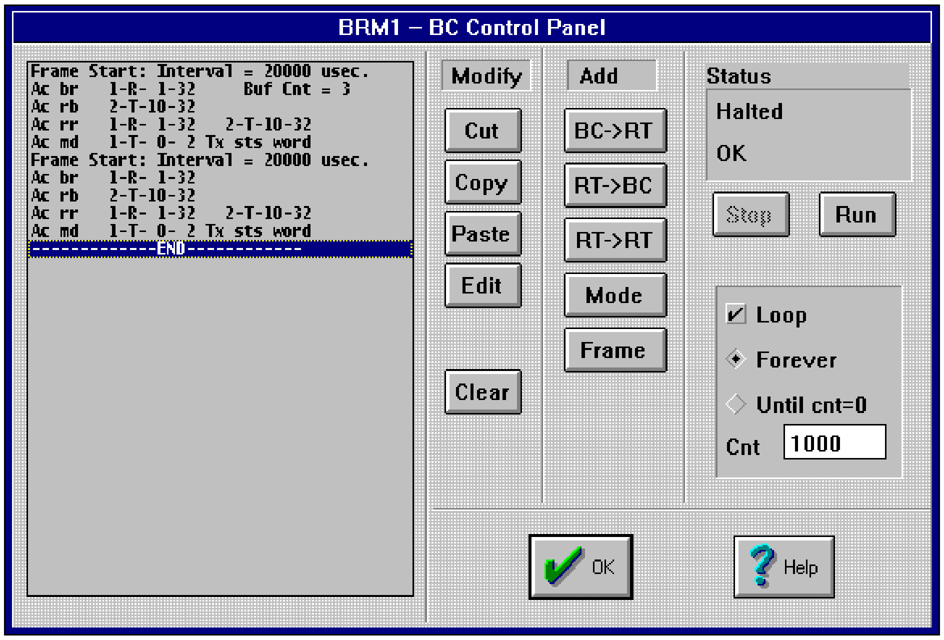
Note: In Windows 95, you can also quit the PASS-1000 application by clicking the close box located in the upper right corner.

3.3 Bus Controller Simulation

The bus controller (BC) controls and coordinates the flow of information on the bus by transmitting commands to remote terminals at predetermined points in time. BC mode of the PASS-1000 consists of a control panel and a set of edit windows that permit a command to be defined and modified and then placed in a bus list. When ready to start bus activity, specify the number of times to execute the bus list and the interval between frames.

3.3.1 BC Control Panel

Choose the BC menu on the Menu Bar to pop up the BC Control Panel (shown in Figure 3.3.1).



```
Frame Start: Interval = 20000 usec.
Ac br   1-R- 1-32     Buf Cnt = 3
Ac rb   2-T-10-32
Ac rr   1-R- 1-32   2-T-10-32
Ac md   1-T- 0- 2 Tx sts word
Frame Start: Interval = 20000 usec.
Ac br   1-R- 1-32
Ac rb   2-T-10-32
Ac rr   1-R- 1-32   2-T-10-32
Ac md   1-T- 0- 2 Tx sts word
-------------END------------
```

Status: Halted, OK

[✓] Loop
[✓ Forever] Until cnt=0
Cnt 1000

Figure 3.3.1: BC Control Panel

Create, view, and execute a bus list from the BC Control Panel. The bus list consists of any combination of bus controller to remote terminal (BC-RT), remote terminal to bus controller (RT-BC), remote terminal to remote terminal (RT-RT), frame start delimiters, and mode commands. The characteristics of any transmitted word of a message may be modified.

The BC Control Panel is divided into three sections from left to right:

- Message List
- Message Editing

➢ Status and Control.

3.3.2 Message List

The message list occupies the left half of the BC Control Panel. This portion of the screen is used to create and edit a list of BC commands. The list is executed when the BC is in run mode. The frame start delimiters are displayed as: Frame Start: Interval = N µs. Each message occupies one line and has the general format of: bus type rt(s). These columns are described in Table 3.3.1.

Table 3.3.1: Message List Format

Column	Description:
bus	This column displays the bus (A or B) that will transmit the command. If a lower case c is displayed to the right of the bus indicator, the BC will execute the next message if there is an error; otherwise, it will halt.
type	The second column contains a two letter designation indicating the type of message transfer. Valid types are: **br** Bus Controller to Remote Terminal **rb** Remote Terminal to Bus Controller **rr** Remote Terminal to Remote Terminal **md** Mode Command
rt(s)	The third column contains the command word(s) that will be transmitted. The format is RT# - T/R - SA# - WC, where RT# and SA# are the remote terminal and subaddress numbers, T/R is Transmit or Receive, and WC is the word count. If the message type is an RT-RT, both the receive and transmit command words will be displayed. If the message is a mode command, an abbreviation for the name of the mode code will directly follow the formatted command word. If the message type is BC-RT and multiple message buffers have been allocated, the number of buffers will be listed as Buf Cnt = N. Also, if errors have been inserted in the command or data words, an asterisk will appear to the left of the command word.

The blue highlight bar is used to indicate a point of action in the bus list. This action may be the insertion or deletion of a message. Move the highlight bar by placing the cursor over the destination point and clicking the left mouse button or by using the Page Up, Page Down, Up Arrow, and Down Arrow keys. If the message list is longer than what can be viewed in the screen, a scroll bar will appear on the right side of the message list permitting the entire list to be viewed. The highlight bar remains in its current location while the list is being browsed.

3.3.3 Message Editing

The middle portion of the BC Control Panel contains push buttons to add, delete, or modify messages in the bus list. To add a message to the bus list, move the highlight bar in the message list to the desired insert location. Place the cursor over a message button, either BC->RT, RT->BC, RT->RT, Mode, or Frame found under the Add title and click the left mouse button. This will pop up a window to define the corresponding message. The new message will appear in the list above the highlight bar. This process may be repeated as many times as necessary to create the desired bus list.

The buttons under the Modify title are used to modify the bus list. The functions of these buttons are described in Table 3.3.2.

Table 3.3.2: Modify Buttons

Button	Use it to:
Cut	Removes the message in the highlight bar from the list and copies it to the paste buffer.
Copy	Copies the message in the highlight bar to the paste buffer.
Paste	Inserts a message from the paste buffer into the list.
Edit	Invokes a window to redefine the corresponding message type.
Clear	Clears the entire message list. A window will appear to verify the action. Click Yes to continue or No to cancel the action.
Ext Input	Adds a “Wait for External Signal” command to the BC List. See the “External Trigger Function” description at the end of the BC Simulation section.

When a message is added to the message list using Add or Paste, the message will be inserted directly above the highlight bar.

While the BC List is running, all of the buttons shown above will be grayed out (inaccessible) with the exception of the Edit button.

Cross Reference: See “Editing Data Buffers While in Run Mode” for more information.

3.3.4 Status and Control

The right portion of the BC Control Panel controls the execution of the bus list and displays status information. The box entitled Status expresses the current state of the BC. The four possible states are described in Table 3.3.3.

Table 3.3.3: Status States

This state:	Means that the BC:
Halted OK	Is halted and no errors occurred during execution of the last message.
Halted *message*	Has halted due to an error. The corresponding error message is displayed in a drop-down box. **Cross Reference:** See Appendix A for further details on error messages.
Run *count*	List is running. A counter is displayed specifying the number of times the BC list executed. It is incremented continuously. An asterisk preceding the count display (**count*) denotes a minor frame execution interval that is too small for proper execution of the messages and data in the frame.
Step	Is currently stepping through a bus list.

The Halted and Run states of the BC are also displayed in the BC text box of the Status Bar. The bus indicator of the Status Bar also indicates bus activity.

The **Loop** check box specifies the number of times to execute the BC list. If this option is not checked, the BC will execute the list one time. If checked, two more check boxes and an edit box appear to permit the user to choose the number of times to execute the BC list. Select the **Forever** check box to run the BC list continuously. Select the **Until cnt=0** check box to run the BC list a specified number of times and enter the number of times in the **Cnt** text box.

Click the Run button to start the BC. If the **Loop** check box is checked, the status indicator will display Running and a count that increments each time the BC list is executed. The BC will continue to run while there are no error conditions that would cause it to halt. To stop the BC, click the Stop button. When the BC is stopped, the status indicator will display Halted - OK. If the Loop check box is not checked, the status box will display Step during execution and then Halted - OK after completion if there were no errors.

3.3.5 Defining a BC-RT or RT-BC Message

Click the BC->RT or RT->BC button in the BC Control Panel to create a bus controller to remote terminal or remote terminal to bus controller message and to modify a word's characteristics. The Define BC->RT Message or Define RT->BC Message window will appear. The Define BC->RT Message window is shown in Figure 3.3.2.

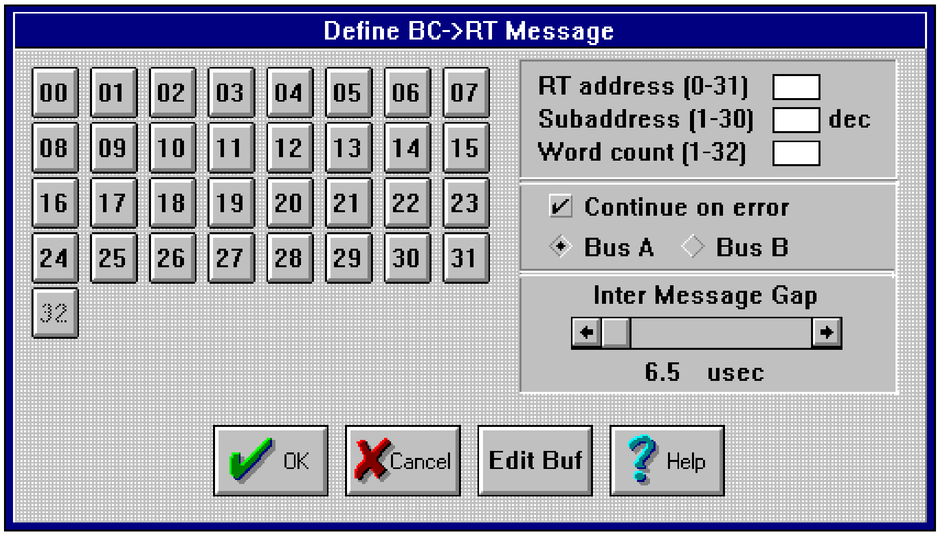


Figure 3.3.2: Message Window

Defining the Command Word

The upper right corner of the window contains three edit boxes that are used to enter the RT address, subaddress, and word count for the message. The edit cursor is always placed in the RT address text box when the window is opened. Select from the number buttons to enter the values for the RT address, subaddress, and word count. These values may also be entered from the keyboard. Entering an invalid number will invoke the User Error window. If this occurs, click OK and enter a valid number to continue.

The button labeled 0 is only available for RT address selection. For 1553B RTs, the 0 and 31 buttons may not be used for a subaddress because they are reserved for mode commands. For 1553A RTs, only subaddress 0 is reserved for Mode commands, subaddress 31 is available for message transmission. RTs are defined as using the 1553B protocol by default. This default setting can be changed by using the **RT Type** selection in the **Tools** pull down menu. Mode commands must be defined from the BC Mode Command window. The 32 button is only available for word count selection since 32 is an invalid remote terminal or subaddress number.

Options Several options are available for modifying the characteristics of the message. These options are described in Table 3.3.4.

Table 3.3.4: Options for Modifying the Message Characteristics

Option	Description
Continue on error	Under normal operation, the bus controller will continue if an error is detected during a message transfer. De-selecting this option forces the BC to stop processing the bus list after an error has been detected.
Bus A or Bus B	Choose one of these option buttons to select the bus over which the message will be transmitted. By default, all messages will be issued on Bus A.
Inter Message Gap	The intermessage gap time may be adjusted by sliding the scroll bar to the desired time. The default intermessage gap is 6.5 microseconds. The gap may be set from 3 to 100 microseconds and is inserted before the message.
Edit Buf	Click this button to pop up the BC->RT Cmd/Data Edit or RT->BC Cmd Edit windows. These windows are used to view, edit, or modify the data and command words for the message.

Editing the BC-RT Command and Data Words Click the Edit Buf button in the Define BC->RT Message window to edit the command and data words of a BC-RT message. The BC->RT Cmd/Data Edit window appears (see Figure 3.3.3).

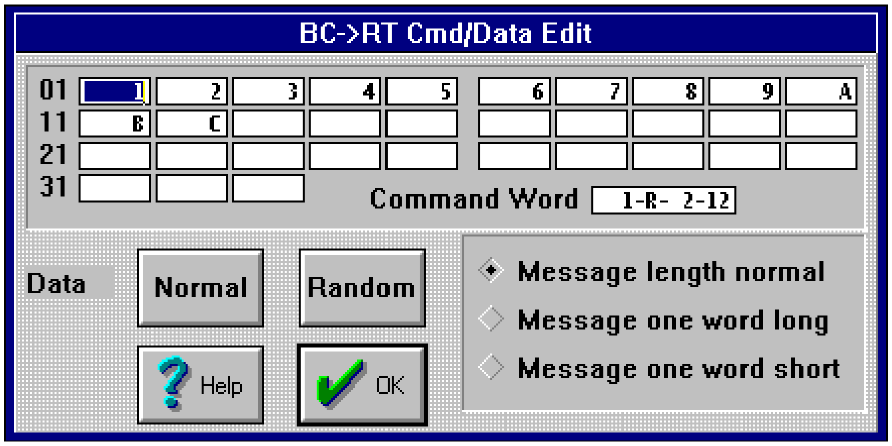


Figure 3.3.3: The BC->RT Command/Data Edit Window

This window contains 33 edit boxes for the data words and one text box for the command word. The 33rd edit box is only required if the Message one word long option is selected. The command word is displayed in the format *RT# - R - SA# - WC*, where *RT#* and *SA#* are the remote terminal and subaddress numbers, *R* is Receive, and *WC* is the word count.

Editing Data Words

There are 33 edit boxes corresponding to each data word that may be transmitted over the bus. Only the number of words specified in the command word may be edited. Each data word is initialized with a value (ranging from 1 to 21 hexadecimal) equal to the order of the word in the buffer. Click the Random button to cause the PASS-1000 to generate random data for the message. Click the Normal button to initialize the data buffer to the original values. Enter specific data values by selecting the desired text box and typing in the data.

To change the characteristics of a data word or the command word, place the cursor over the desired word and click the *right* mouse button. This will invoke the Transmit Word Options window.

Cross Reference: For more information, see the section *Transmit Word Options*.

Message Length Options

Three option buttons control the number of words which will be transmitted. These buttons are described in Table 3.3.5.

Table 3.3.5: Message Length Option Buttons

Option	Description
Message length normal	The number of words transmitted follows the word count specified in the command word.
Message one word long	The number of words transmitted will be one greater than the word count specified in the command word. The additional data word will appear in the corresponding edit box.
Message one word short	The number of words transmitted will be one less than the word count specified in the command word. The last data word will be removed from the corresponding edit box.

Buffer Management

For BC-RT commands, multiple buffers can be allocated. Buffers are added after the current buffer using the **Add** button. The current buffer can be deleted by clicking on the **Delete** button as long as additional buffers remain for the BC-RT message. The last remaining buffer for any BC-RT message can only be deleted by deleting the BC-RT message from the message list. A specific buffer can be viewed or edited by typing the number of the desired buffer in the **Buffer _ of N** box and pressing Enter or Tab.

The BC-RT buffers created at this level are linked together in a circular list. On each repetition of the entire BC message list, the current message buffer for the BC-RT message is transmitted and then replaced by the next buffer in the circular buffer list for that BC-RT. When the last buffer in the circular list is transmitted, it is replaced by the first buffer in the list resulting in a continuous progression. This allows the data or errors transmitted with a given BC-RT command to be varied on subsequent repetitions of a BC message list.

Editing the RT-BC Command Word

Click the Edit Buf button in the Define RT->BC Message window to edit the command word of an RT-BC message. The RT->BC Cmd Edit window appears (see Figure 3.3.4).

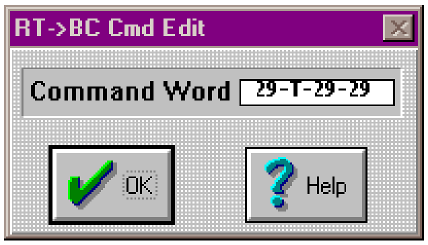


Figure 3.3.4: RT->BC Cmd Edit Window

The RT->BC Cmd Edit window displays the command word that will be sent over the bus. The command word is displayed in the format *RT#* - *T* - *SA#* - *WC*, where *RT#* and *SA#* are the remote terminal and subaddress numbers, *T* is Transmit, and *WC* is the word count. To change the characteristics of the command word, place the cursor over the command word and click the *right* mouse button. This will invoke the Transmit Word Options window.

Cross Reference: For more information, see the section *Transmit Word Options.*

3.3.6 Defining an RT-RT Message

Click the RT->RT button in the BC Control Panel to create an remote terminal to remote terminal message and to modify a word's characteristics. The Define RT->RT Message window appears (see Figure 3.3.5).

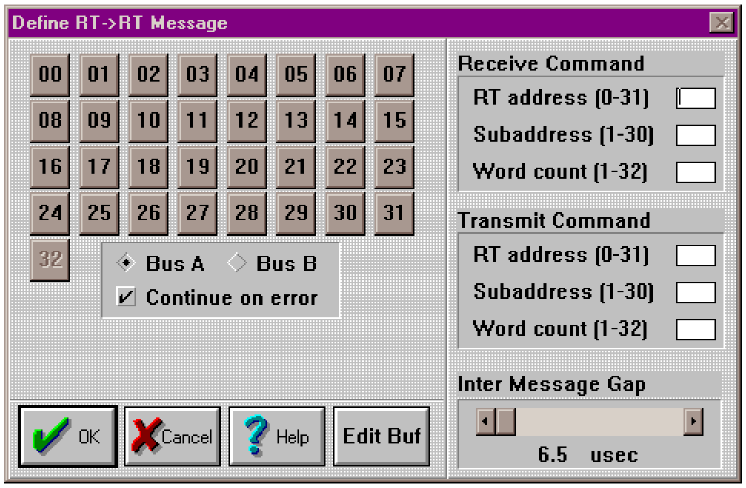


Figure 3.3.5: Define RT->RT Message Window

Defining the Command Words

The right side of this window contains two groups of three edit boxes that are used to enter the RT address, subaddress, and word count for the receive and transmit command words. The edit cursor is always placed in the RT address text box when the window is opened. Select from the number buttons to enter the values for the RT address, subaddress, and word count. The values may also be entered from the keyboard. Entering an invalid number will invoke the User Error window. If this occurs, click OK and enter a valid number to continue.

The button labeled 0 is only available for RT address selection. For RTs using the 1553B protocol, the 0 and 31 buttons may not be used for a subaddress because they are reserved for mode commands. RTs using the 1553A protocol may use subaddress 31. All RTs are defined as using the 1553B protocol by default but can be changed to use the 1553A protocol by using the **RT Type** dialog in the **Tools** pull down menu. Mode commands must be defined from the BC Mode Command window. The 32 button is only available for word count selection since 32 is an invalid remote terminal or subaddress number.

Options

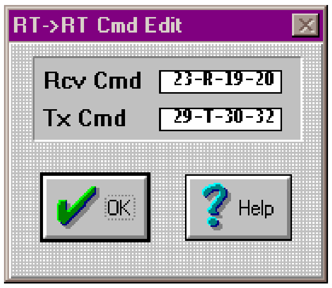
Several options are available for modifying the characteristics of the message. These options are described in Table 3.3.6.

Table 3.3.6: Options for Modifying the Message Characteristics

Option	Description
Continue on error	Under normal operation, the bus controller will continue if an error is detected during a message transfer. De-selecting this option forces the BC to stop processing the bus list after an error has been detected.
Bus A or Bus B	Choose one of these option buttons to select the bus over which the message will be transmitted. By default, all messages will be issued on Bus A.
Inter Message Gap	The intermessage gap time may be adjusted by sliding the scroll bar to the desired time. The default intermessage gap is 6.5 microseconds. The gap may be set from 6.5 to 100 microseconds and is inserted before the message.
Edit Buf	Click this button to pop up the RT->RT Cmd/Data Edit window. This window is used to view, edit, or modify the command words for the message.

Editing the RT-RT Command Words

Click the Edit Buf button in the Define RT->RT Message window to edit the command words of an RT-RT message. The RT->RT Cmd Edit window appears (see Figure 3.3.6).



23-R-19-20

29-T-30-32

Figure 3.3.6: RT->RT Cmd Edit Window

The RT->RT Cmd Edit window displays the transmit and receive command words that will be sent over the bus. The command words are displayed as *RT#* - *T/R* - *SA#* - *WC*, where *RT#* and *SA#* are the remote terminal and subaddress numbers, *T/R* is Transmit or Receive, and *WC* is the word count. To change the characteristics of a command word, place the cursor over the command word and click the right mouse button. This will invoke the Transmit Word Options window.

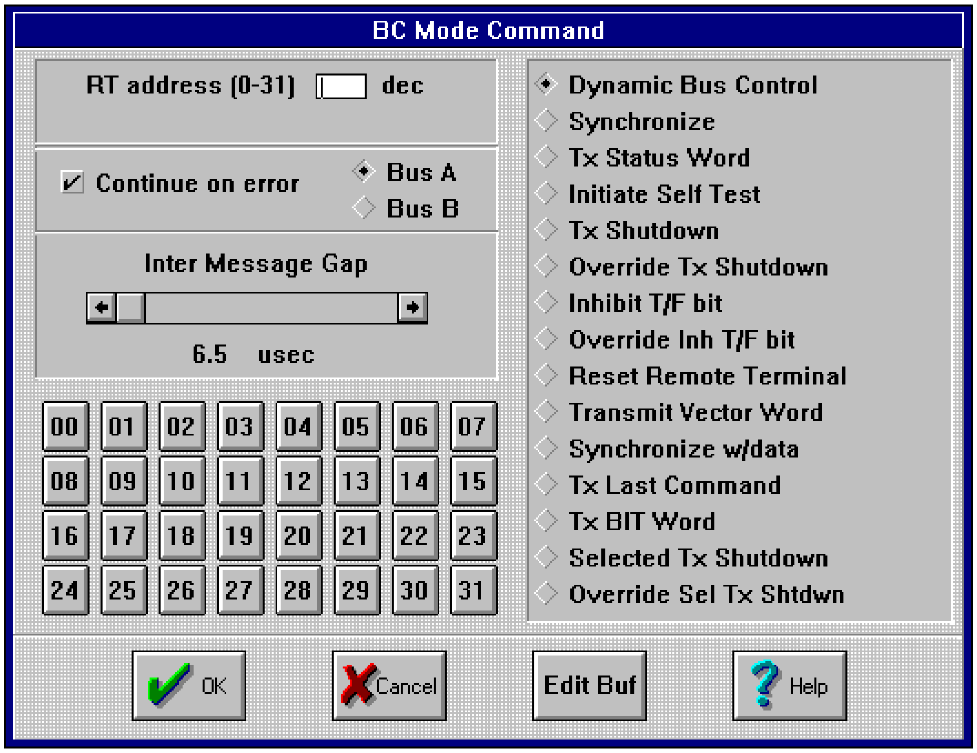
Cross Reference: For more information, see the section *Transmit Word Options*.

3.3.7 Defining a Mode Command Message

Click the Mode button in the BC Control Panel to create a mode command and to modify a word's characteristics. The upper left corner of the BC Mode Command window contains an edit box that is used to enter the RT address. The edit cursor is always placed here when the window is opened. Enter a value for the RT address using either the number buttons or the keyboard. Entering an invalid number will invoke the User Error window. If this occurs, click OK and enter a valid number to continue.

1553B mode commands can use either subaddress 0 or subaddress 31. 1553A protocol restricts the user to subaddress 0 for mode commands.

If a 1553B RT is chosen, the BC Mode Command window appear (see Figure 3.3.7).



BC Mode Command

RT address (0-31) [ ] dec

[✓] Continue on error

[✓ Bus A]
Bus B

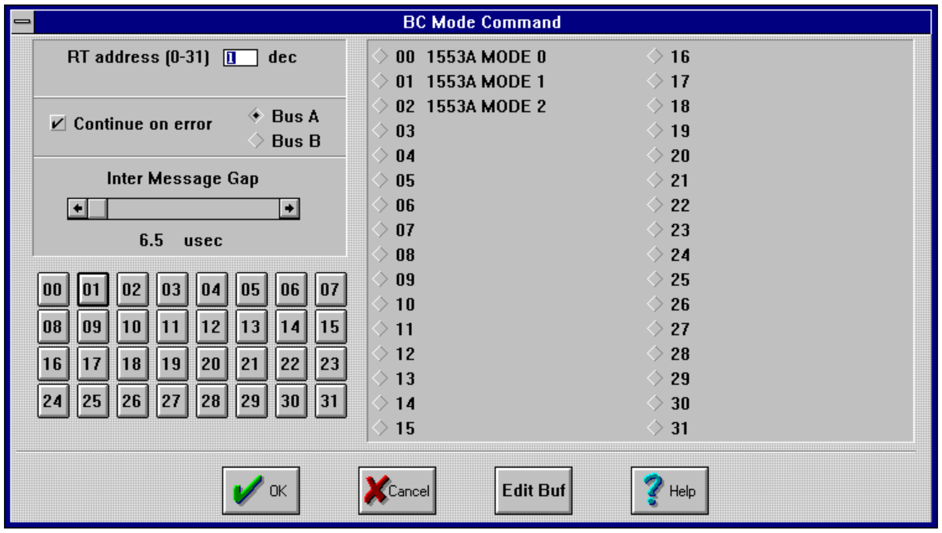
Inter Message Gap
6.5 usec

00 01 02 03 04 05 06 07
08 09 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31

[✓ Dynamic Bus Control]
Synchronize
Tx Status Word
Initiate Self Test
Tx Shutdown
Override Tx Shutdown
Inhibit T/F bit
Override Inh T/F bit
Reset Remote Terminal
Transmit Vector Word
Synchronize w/data
Tx Last Command
Tx BIT Word
Selected Tx Shutdown
Override Sel Tx Shtdwn

Figure 3.3.7: BC Mode Command Window

This window is replaced by the window shown in Figure 3.3.8 if a 1553A RT is chosen. (1553A protocol is assigned to an RT using the **RT Type** selection in the **Tools** pull down menu.) The 1553A window will display the available 1553 mode codes with any associated description provided by the user in the **Mode Definition** dialog (See **Utilities** Section).



BC Mode Command

RT address (0-31) [1] dec

[✓] Continue on error

[✓ Bus A]
Bus B

Inter Message Gap
6.5 usec

00 01 02 03 04 05 06 07
08 09 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31

00 1553A MODE 0	16
01 1553A MODE 1	17
02 1553A MODE 2	18
03	19
04	20
05	21
06	22
07	23
08	24
09	25
10	26
11	27
12	28
13	29
14	30
15	31

Figure 3.3.8: BC Mode Command Window

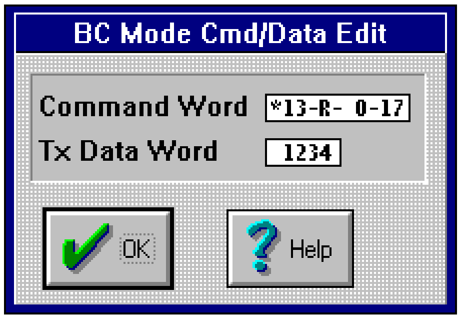
The right half of all of the Mode Command windows lists the types of mode codes. Select one to define using the mouse. If there is an associated data word for the mode command, a data word text box will appear beneath the RT address text box (upper left corner of the screen). Enter the desired value for the data word in hexadecimal.

Options Several options are available for modifying the characteristics of the message. These options are described in Table 3.3.7.

Table 3.3.7: Options for Modifying the Message Characteristics

Option	Description
Sub Address 0 or 31	For 1553B ONLY, select the subaddress to be used for the mode command.
Continue on error	During normal operation, the bus controller will continue if an error is detected during a message transfer. De-select this option to force the BC to stop processing the bus list after an error has been detected.
Bus A or Bus B	Choose one of these option buttons to select the bus over which the message is to be transmitted. By default, all messages will be issued on Bus A.
Inter Message Gap	Adjust the intermessage gap time by sliding the scroll bar to the desired time. The default intermessage gap is 6.5 microseconds. The gap may be set from 6.5 to 100 microseconds and is inserted before the message.
Edit Buf	Click this button to bring up the BC Mode Cmd/Data Edit window. Use this window to view, edit, or modify the command and data word for the message.

Editing the Mode Command and Data Words Click the Edit Buf button in the BC Mode Command window to view or modify the mode command. The BC Mode Cmd/Data Edit window appears (see Figure 3.3.9).



Command Word *13-R- 0-17
Tx Data Word 1234

Figure 3.3.9: BC Mode Cmd/Data Edit Window

The BC Mode Cmd/Data Edit window displays the command word that will be sent on the bus. The command word is displayed as *RT# - T - SA# - MC*, where *RT#* and *SA#* are the remote terminal and subaddress numbers, *T/R* is Transmit or Receive, and *MC* is the mode code. If the mode command has an associated data word, it will also be displayed. To change the characteristics of the command or data word, place the cursor over the word's text box and click the *right* mouse button. This will invoke the Transmit Word Options window. For more information, see the section "Transmit Word Options".

3.3.8 Defining a Minor Frame

The Frame button allows the message list to be broken up into minor frames. A minor frame is defined as the messages between two successive Frame Start entries. The interval time for each minor frame is set in the **Frame Start** dialog which appears when you press the **Frame** button (see Figure 3.3.10).

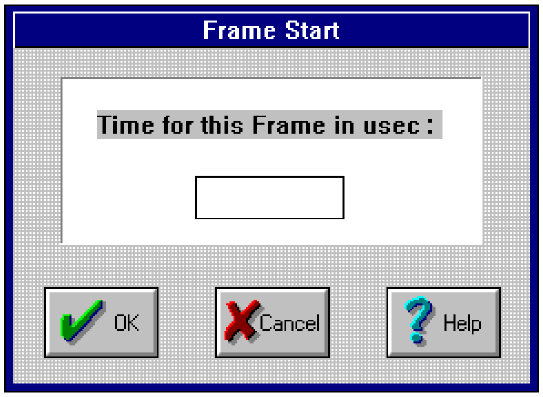


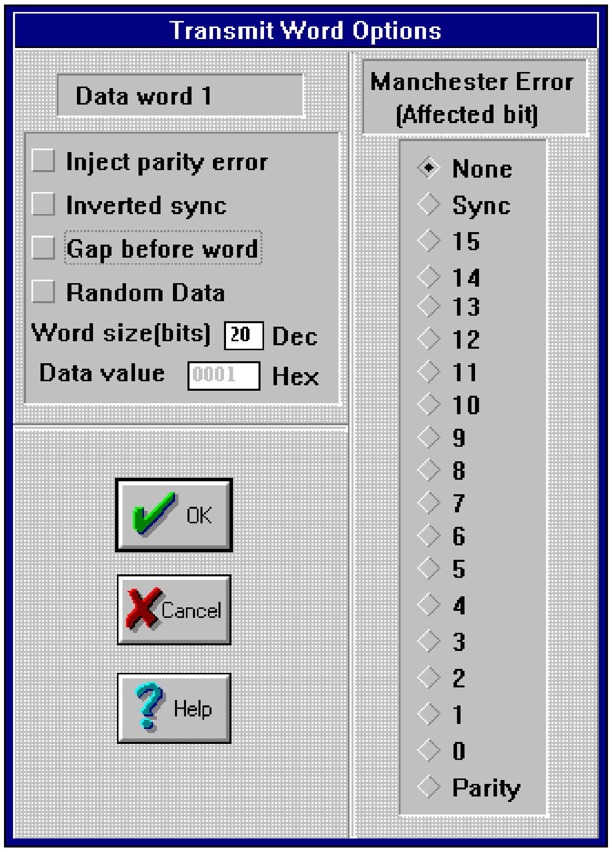
Figure 3.3.10: Frame Start Dialog

The default interval value is 20,000 microseconds. When a Frame Start message is encountered during processing, a countdown timer is set with the interval value. At the end of the frame (determined by the next Frame Start message), transmission of the message list is halted until the timer reaches zero. This assures a time difference of at least the interval value between the start of one minor frame and the start of the next. If the interval value for a particular minor frame is less than that required by the messages in the frame, an asterisk will appear to the left of the displayed run count.

By default, each message list starts with a Frame Start message. The first Frame Start can be edited or copied but not deleted. All additional Frame Start entries in the message list can be copied, deleted, or edited in the same manner as the command messages in the list.

3.3.9 Transmit Word Options

This window, shown in Figure 3.3.11, appears whenever the characteristics of a transmitted word are modified from the RT Receive Response Edit, RT Transmit Response Edit, RT Mode Response Edit, BC->RT Cmd/Data Edit, RT->BC Cmd Edit, RT->RT Cmd Edit, or BC Mode Cmd/Data Edit windows.


Transmit Word Options

Data word 1

- [ ] Inject parity error
- [ ] Inverted sync
- [ ] Gap before word
- [ ] Random Data

Word size(bits) 20 Dec

Data value 0001 Hex

Manchester Error (Affected bit)

[✓ None] Sync 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 Parity

Figure 3.3.11: Transmit Word Options Window

A window in the upper left corner displays which word is being changed. While a word is in this window, its value may not be changed. It is displayed in the **Data value** box for reference purposes only. The value may only be changed from the window that opened the Transmit Word Options window.

Two to four check boxes may be displayed (the number depends upon the type of word being modified) to permit the insertion of certain errors into a word. A description of each check box appears in Table 3.3.8.

Table 3.3.8: Check Boxes Permitting Insertion of Errors into a Word

Check Box	Description
Inject parity error	The parity is inverted for the word.
Inverted sync	The word is transmitted with the wrong sync type. A data word will be given a command sync while a status or command word will be given a data sync.
Gap before word	A gap of approximately three microseconds is inserted before the data word. This option is available for data words only.
Random Data	The data word is filled with a random value. This option is available for data words only.

The number of bits in the word may be changed by modifying the value in the **Word size(bits)** text box. The right half of the window permits insertion of errors into the Manchester Encoding of the word. Choose the bit position in the data word which will have the inserted Manchester error. The midpoint crossing will be deviated –250 nanoseconds.

3.3.10 External Trigger function

The External Trigger function suspends BC list transmission until a signal is received on the TRIGGER line of the PASS cable adapter. The PASS performs this function when it encounters a “Wait for External Signal” command in the BC list. The PASS waits for the external signal and then transmits the remaining commands in the BC list. If the BC list is configured in a loop or if multiple “wait” commands are inserted in a single list, processing is suspended each time a wait command is encountered. Use the Stop button to end BC processing during a wait, if necessary.

Insert a Wait for External Signal command into a BC list using the Ext Input button on the BC Control Panel. To add a Wait for External Signal command, move the highlight bar to the desired location in the BC list and click on the Ext Input button. The command will appear in the list above the highlight bar. Wait for External Signal commands can only be added or deleted; they cannot be edited or copied.

3.3.11 Editing Data Buffers While in Run Mode

BC-RT data buffers may be edited while the BC List is running. To edit buffers, highlight the desired BC-RT message and click the Edit button. If more than one buffer is configured for the message, the Select BC-RT Data Buffer window will appear. Enter the number of the buffer you wish to edit and click OK. The Edit BC-RT Data window will appear.

This window contains 33 edit boxes for the data words. The 33rd edit box is only required if the **Message one word long** option is selected. Click the Random button to cause the PASS-1000 to generate random data for the message. Click the Normal button to initialize the data buffer to the original values. Enter specific data values by selecting the desired text box and typing in the data. The characteristics of a data word cannot be altered while the BC List is running.

3.4 Remote Terminal Simulation

Remote Terminal (RT) mode may be used to emulate one or more RTs or to view activity in an RT receive buffer. Choose from two modes of operation - Define or View. In Define mode, activate RTs and subaddresses from a series of windows and define how the PASS-1000 should respond to commands received by the RT. In View mode, display the data in an RT receive buffer in real-time. An RT may operate in both modes simultaneously.

3.4.1 Enabling and Disabling Rts

From the RT menu, choose Define. The RT Enable/Disable window appears (see Figure 3.4.1).

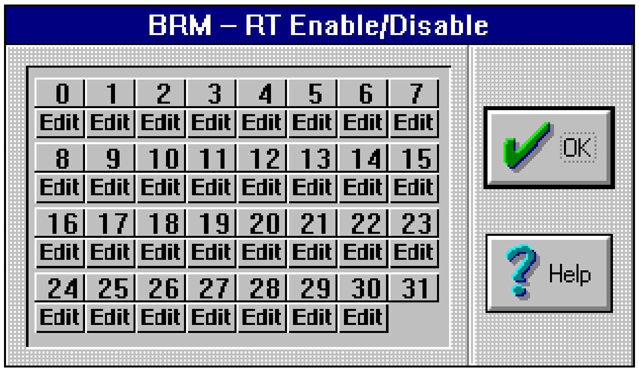

BRM – RT Enable/Disable
0 1 2 3 4 5 6 7
Edit Edit Edit Edit Edit Edit Edit Edit
8 9 10 11 12 13 14 15
Edit Edit Edit Edit Edit Edit Edit Edit
16 17 18 19 20 21 22 23
Edit Edit Edit Edit Edit Edit Edit Edit
24 25 26 27 28 29 30 31
Edit Edit Edit Edit Edit Edit Edit
OK
Help

Figure 3.4.1: RT Enable/Disable window

This window contains a grid of push-button pairs: one pair for each possible RT address. For 1553B RTs, there are 31 push-button pairs; for 1553A RTs, there are 32 pairs. The upper button of each pair is used to turn the RT on or off and to indicate the current mode and status of the RT. The lower half of the button opens the next window level which is used to enable subaddresses and modify data and status words. To enable or disable a specific RT, select the upper half of the desired RT push button. This action toggles the button up or down. Up (light gray) indicates that the RT is disabled. Down (dark gray) indicates that the RT is enabled. The status of the RT is expressed with the colors listed in Table 3.4.1.

Table 3.4.1: RT Status Colors

Color	Description
Light Gray	The button is up, indicating that the associated RT is not enabled. There are no subaddress buffers allocated to this RT.

Color	Description
Dark Gray	The button is down, indicating the associated RT is enabled. No buffers have been allocated to any subaddresses. In this mode, the RT will respond to mode commands but not to commands directed at data subaddresses.
Yellow	The button is up, indicating that the associated RT is not enabled. There are subaddress buffers allocated to this RT.
Green	This button is down, indicating that the associated RT is enabled. There are subaddress buffers allocated to this RT.

Note: When an RT is not enabled, it may still have subaddress buffers allocated. Data buffers consume memory. If these buffers are no longer needed, it is best to de-allocate them so that the memory may be used elsewhere.

For 1553B RTs only: RT 31 is reserved for broadcast commands. Subaddress buffers cannot be allocated.

3.4.2 (De)Allocation of RT Subaddresses

Choose an Edit button from the RT Enable/Disable window to allocate subaddress data buffers. The RT Subaddress (De)Allocate window appears (see Figure 3.4.2).

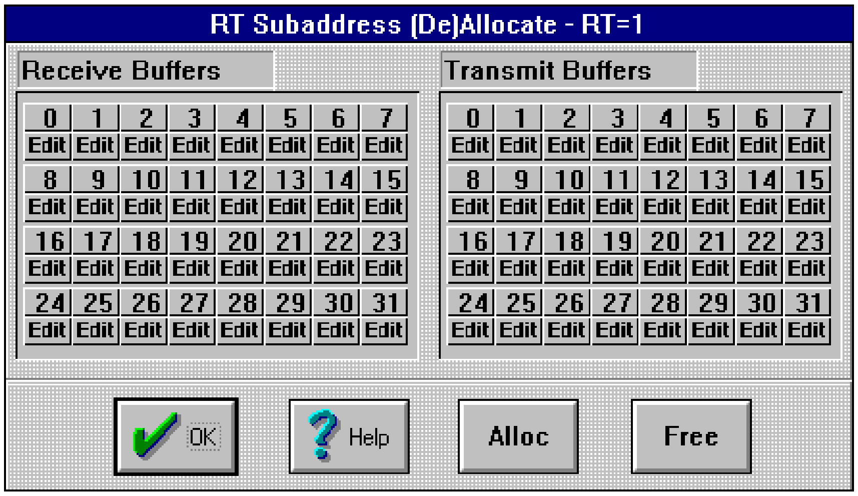


Figure 3.4.2: RT Subaddress (De)Allocate window

This window contains a grid of 64 push-button pairs: one for each possible subaddress. The upper half of each button is used to allocate or de-allocate data buffers for the associated subaddress. If the RT is defined as 1553B (default), four subaddresses are reserved for mode commands and do not require allocated buffers. These subaddresses include SA0 receive and transmit and SA31 receive and transmit. If the RT is defined as 1553A, only subaddresses SA0 receive and transmit are reserved for mode commands. The remaining subaddresses are allowed data buffers. Whenever a subaddress has an allocated data buffer, the PASS-1000 recognizes it as a legal subaddress and responds with a status word. Each allocated buffer uses approximately 58 words of PASS-1000 memory.

Choosing the upper half of one of these push buttons toggles it down and allocates a buffer for that subaddress. The button displays the state of the subaddress with the colors listed in Table 3.4.2.

Table 3.4.2: Subaddress State Colors

Color	Description
Light Gray	No buffer is associated with the subaddress.
Green	A buffer has been allocated and no errors are associated with it.
Red	A buffer has been allocated and an error(s) has been inserted on a data or status word(s). Click this button to quickly clear the errors.

Two additional buttons, described in Table 3.4.3, are provided for convenience.

Table 3.4.3: Additional Subaddress Allocation Buttons

Button	Description
Alloc	Allocates buffers to all subaddresses.
Free	Deallocates all subaddress buffers and frees memeory

Each subaddress push-button has a corresponding Edit button. This button opens a window to edit the contents of a subaddress's buffer. A subaddress buffer cannot be edited if the upper half of the button has not been toggled to "on" (the color will be light gray). This button will be colored to indicate an allocated buffer when the edit is complete. If a subaddress buffer has been allocated and edited, the contents of that buffer will be lost if the buffer is de-allocated either with the Free button or by toggling the individual subaddress buttons from "on" to "off".

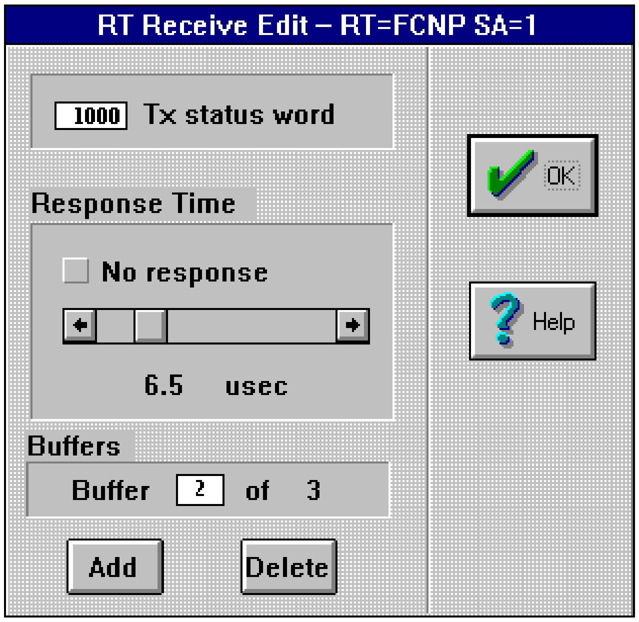
The PASS-1000 will respond to commands to a subaddress when the RT is enabled and a buffer is allocated to that subaddress. If the RT is disabled or no subaddress buffer is allocated, the PASS-1000 will not respond. Mode commands do not require an explicit allocation of buffers. When the RT is enabled, the PASS-1000 will respond to all mode commands. The subaddresses associated with mode commands are 0 and 31 for 1553B and 0 for 1553A. These buttons are green when the RT is enabled and gray when it is disabled.

For receive and transmit subaddresses, multiple message buffers can be allocated. The buffers for any RT-SA are linked together in a circular list with the current buffer advancing on each subsequent transmission to the RT-SA. This allows the data and status word transmitted by each RT-SA to be updated during a transmission cycle.

Three editing windows are available for modifying the response to receive, transmit, and mode commands. Whenever the characteristics of a word are changed or a no response is selected, the subaddress's push button in the RT Subaddress (De)Allocate window will turn red to indicate that an error has been inserted. Click this button to clear the error(s).

Editing the Response of an RT Receive Command

To modify the status word of a receive subaddress, choose an Edit button from the Receive Buffers window. The RT Receive Response Edit window appears (see Figure 3.4.3).


RT Receive Edit – RT=FCNP SA=1

1000 Tx status word

Response Time

No response

6.5 usec

Buffers

Buffer 2 of 3

Figure 3.4.3: RT Receive Response Edit Window

This window permits modification of the status word for a subaddress: the only word transmitted in response to a receive command word. The value of the status response word may be changed by editing its text box. To change the characteristics of the status word, place the cursor over the text box and click the *right* mouse button. This will invoke the Transmit Word Options window.

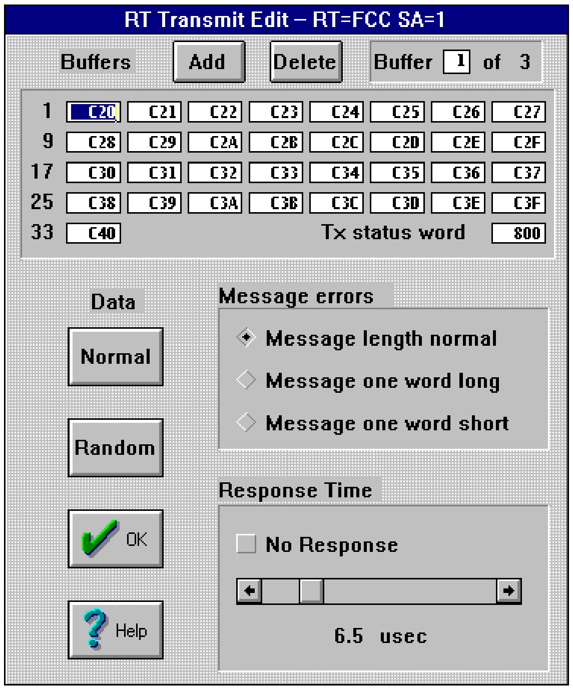
Cross Reference: For more information, see the section *Transmit Word Options.*

The status word response time may be adjusted by sliding the scroll bar to the desired time. The default response time is 6.5 microseconds. The response may be set from 3 to 20 microseconds. If no response is desired, select the `no response` check box. The scroll bar disappears when this box is checked.

Additional buffers are added by clicking on the **Add** button. The status word and response time can be modified in each allocated buffer as described above. Any added buffer can be deleted by clicking on the Delete button. The last buffer for the RT-SA cannot be deleted using this screen. The last buffer must be deallocated using the **Subaddress (De)Allocate** dialog. To access a particular buffer, type the desired buffer number in the **Buffer _ of N** edit box and Enter or Tab.

Editing the Response of an RT Transmit Command

To modify the status word and data buffer of a transmit subaddress, choose an Edit button from the Transmit Buffers window. The RT Transmit Response Edit window appears (see Figure 3.4.4).


RT Transmit Edit – RT=FCC SA=1

Buffers | Add | Delete | Buffer 1 of 3

1	C20	C21	C22	C23	C24	C25	C26	C27
9	C28	C29	C2A	C2B	C2C	C2D	C2E	C2F
17	C30	C31	C32	C33	C34	C35	C36	C37
25	C38	C39	C3A	C3B	C3C	C3D	C3E	C3F
33	C40						Tx status word	800

Data: Normal | Random | OK | Help

Message errors
- [✓ Message length normal]
- Message one word long
- Message one word short

Response Time
- No Response
- 6.5 usec

Figure 3.4.4: RT Transmit Response Edit Window

This window displays edit boxes for 33 data words and the status word for the transmit subaddress, plus options to inject message errors, adjust the status word response time, and add additional message buffers. The contents and characteristics of any word may be modified. Each data word is initialized with a value equal to the command word for the subaddress. (The word count field is equal to the order of the data word in the buffer, with data word one being equal to zero.) Click the Random button to cause the PASS-1000 to generate random data for the message. Click the Normal button to initialize the data buffer to the original values. Enter specific data values by selecting the desired text box and typing in the data.

To change the characteristics of a word, place the cursor over the text box and click the *right* mouse button. This will invoke the Transmit Word Options window. For more information, see the section "Transmit Word Options". If the characteristics of a word are changed from the default values, an asterisk appears beside the word.

Use the Message errors box to inject a Low Word Count or High Word Count error. Choosing Message one word long will cause the RT to transmit one word more than the value in the word count field of the transmit command word. Choosing Message one word short will cause the RT to transmit one word less than the requested word count. Data word 33 may be modified if Message one word long is chosen.

The status word response time may be adjusted by sliding the scroll bar to the desired time. The default response time is 6.5 microseconds. The response may be set from 3.5 to 20 microseconds. If no response is desired, select the No response check box. The scroll bar disappears when this box is checked.

Additional buffers are added by clicking on the **Add** button. The status word, data words, message length, and response time can be modified in each allocated buffer as described above. Any added buffer can be deleted by clicking on the **Delete** button. The last buffer for the RT-SA cannot be deleted using this screen. The last buffer must be deallocated using the **Subaddress (De)Allocate** dialog. To access a particular buffer, type the desired buffer number in the **Buffer _ of N** edit box and Enter or Tab.

Editing the Response of an RT Mode Command

To modify the status or data word of a mode command for a 1553B RT, choose the subaddress 0 or 31 Edit button from either the Receive Buffers window or Transmit Buffers window. The RT Mode Response Edit window shown in Figure 3.4.5 appears.

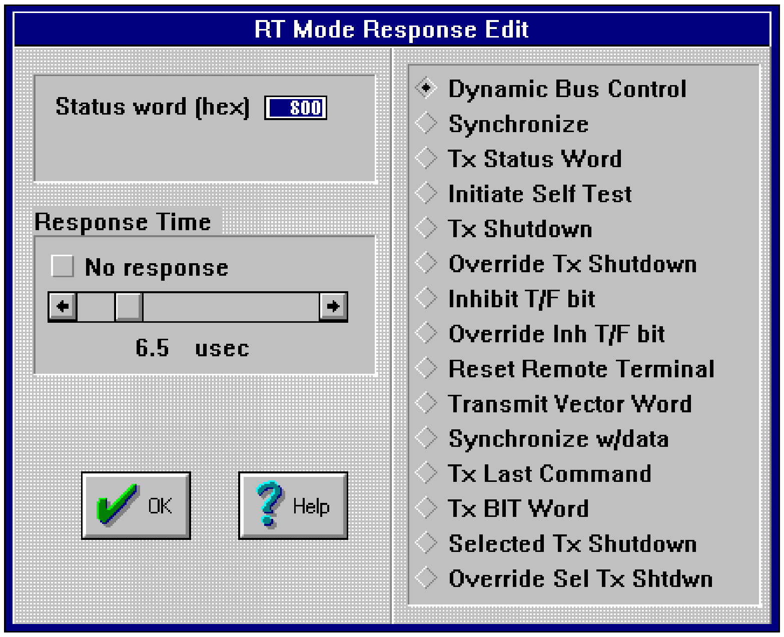


Figure 3.4.5: RT Mode Response Edit Window

This window displays each of the fifteen possible mode codes and their status words. The contents and characteristics of each status word and data word, where applicable, may be modified.

Select the desired mode code using one of the option buttons on the right. The status word for that mode command will be displayed in hexadecimal in the **Status word** text box. If **Transmit Vector Word** or **Transmit BIT Word** are selected, a text box for the data word will appear. Enter specific values for the status response or data word by selecting the desired text box and typing in the data.

The screen shown in Figure 3.4.6 appears for 1553A RTs:

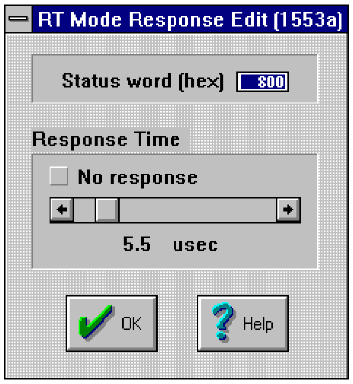


Figure 3.4.6: RT Mode Response Edit Window

All 1553A mode codes share the same status word and response time, values which are set in the above dialog. Change the status word by selecting the edit box and typing a new value.

To change the characteristics of a word for either protocol, place the cursor over the text box and click the *right* mouse button. This will invoke the Transmit Word Options window. For more information, see the section "Transmit Word Options". If the characteristics of a word are changed from the default values, an asterisk will appear beside the affected word.

Adjust the status word response time in either dialog by sliding the scroll bar to the desired time. The default response time is 6.5 microseconds for 1553B RTs and 5.5 microseconds for 1553A RTs. The response may be set from 3 to 20 microseconds. If no response is desired, select the **No response** check box. The scroll bar disappears when this box is checked.

3.4.3 RT Buffer Viewing

The RT View feature permits the inspection of multiple RT subaddress data buffers (as many as memory will allow) while they are transmitting or receiving data. This feature is generally used to monitor a specific RT buffer in real time. From the RT Menu, select View. The View RT Select menu shown in Figure 3.4.7 appears.

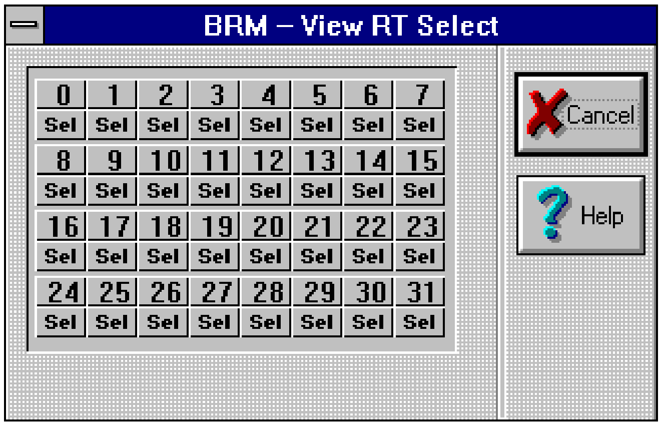


Figure 3.4.7: View RT Select Menu

This window displays a grid of 32 push buttons used to select a desired RT for viewing. Above each button is the corresponding RT number. To select an RT for viewing, click the Sel button of the RT. This will open the View SA Select window (shown in Figure 3.4.8) from which a transmit or receive subaddress may be selected.

Note: Transmit subaddresses that are emulated by the PASS-1000 may not be viewed. Only transmit subaddresses that are emulated external to the PASS-1000 may be viewed.

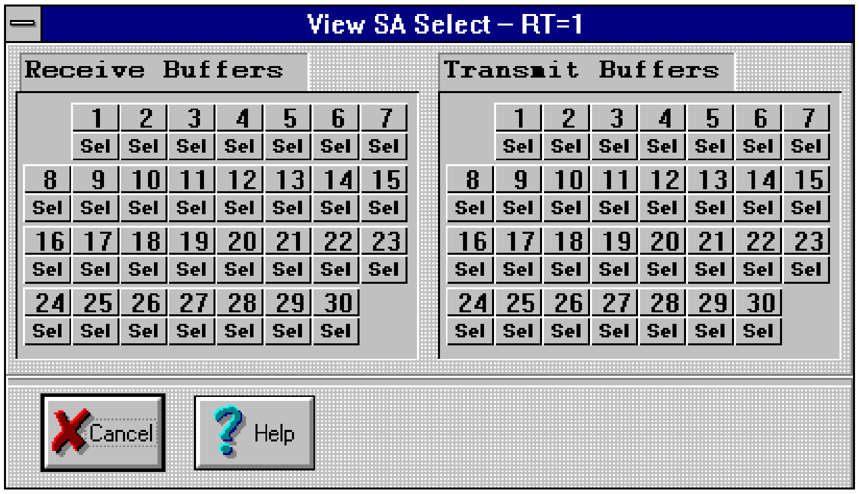
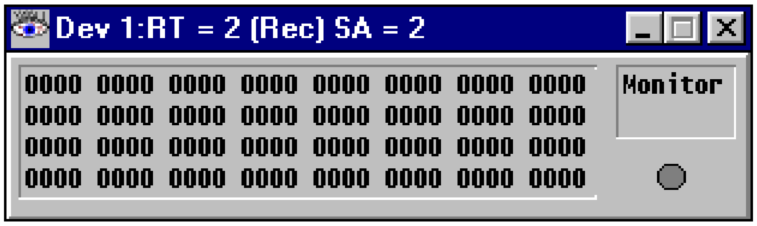


Figure 3.4.8: View SA Select Window

This window displays two grids of 30 push buttons representing 30 transmit and 30 receive data subaddresses. Mode codes may not be viewed. To view a subaddress buffer, click the Sel button of the desired subaddress. This action will close the View RT Select panel and View SA Select panel and open the view window for the remote terminal and subaddress selected (see Figure 3.4.9).



```
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
```

Figure 3.4.9: RT View Window

The RT View window displays all 32 data words of a subaddress buffer in real-time. Data words are displayed in hexadecimal. The RT and subaddress numbers and information about whether the subaddress is to transmit or receive data are displayed in the Title Bar of the window. A status box, displayed on the right side of the window, specifies one of the two subaddress states described in Table 3.4.4.

Table 3.4.4: Subaddress States

State	Description
Emulate	The subaddress selected for viewing is currently emulated in the PASS-1000.
Monitor	The subaddress selected for viewing is not currently emulated in the PASS-1000. This provides a means of viewing the data being transmitted or received by an external RT.

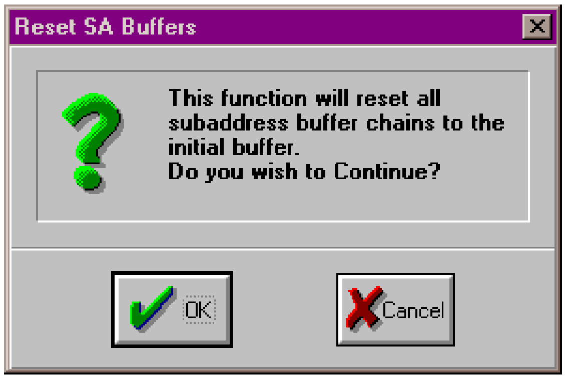
A circular activity indicator turns green if there is current activity in the buffer; it is gray otherwise. A word count entry specifies the number of valid words received by the subaddress. It is displayed in the status box as **WC : x**, where **x** is the number of words.

Three data buffers are allocated when an RT-SA view window is opened. The buffers are linked in a circular list and are updated in order. The view window is updated approximately every half second and will show the most recently updated data buffer. The buffers used by the view windows are allocated from the same memory area on the ABI-PASS that is used by BC list processing.

Multiple RT View windows may be active at one time to view multiple buffers. They may be minimized to an icon by clicking the down arrow box in the upper right corner of the window. The icon will usually appear in the lower left portion of the screen. Double-click the icon to pop up the window for viewing. To close the window, click the Control-box in the upper left corner of the window then choose **Close**.

3.4.4 Reset SA Buffers

If multiple data buffers are allocated for one or more subaddresses and execution is halted in the middle of a list, this function may be used to reset all subaddress buffer chains to the inital buffer. Select Reset SA Buffers from the RT menu to open the window shown in Figure 3.4.10.



This function will reset all subaddress buffer chains to the initial buffer.
Do you wish to Continue?

Figure 3.4.10: Reset SA Buffers Window

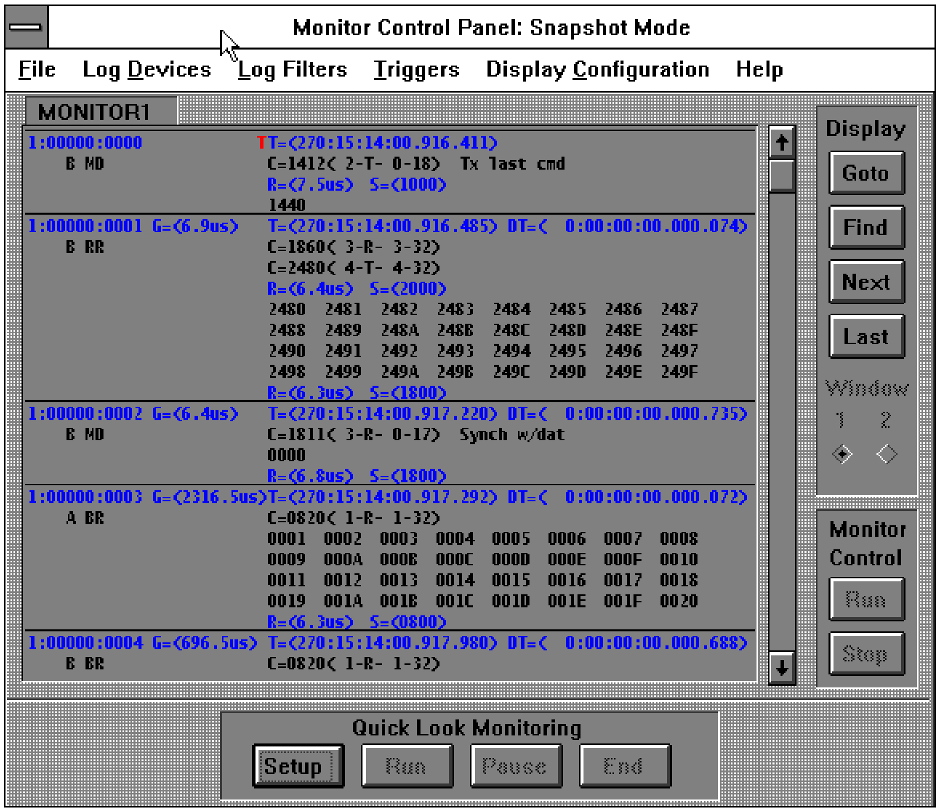
Select **OK** to reset all subaddress buffer chains to the initial data buffer.

3.5 Bus Monitoring

Monitor mode may be used to monitor traffic on a MIL-STD-1553 bus. There are two modes of monitor operation: Snapshot and Data Logging. Both modes allow the user to monitor data, filter messages, trigger on specific messages, and examine the data; however, Data Logging mode automatically stores the data in a user defined binary file. This section discusses Snapshot mode.

3.5.1 Monitor Control Panel: Snapshot Mode

From the Monitor menu, choose Snapshot Mode. The Monitor Control Panel shown in Figure 3.5.1 will appear.



```
1:00000:0000              TT=(270:15:14:00.916.411)
    B MD                   C=1412( 2-T- 0-18)  Tx last cmd
                           R=(7.5us)  S=(1000)
                           1440
1:00000:0001 G=(6.9us)     T=(270:15:14:00.916.485) DT=(  0:00:00:00.000.074)
    B RR                   C=1860( 3-R- 3-32)
                           C=2480( 4-T- 4-32)
                           R=(6.4us)  S=(2000)
                           2480  2481  2482  2483  2484  2485  2486  2487
                           2488  2489  248A  248B  248C  248D  248E  248F
                           2490  2491  2492  2493  2494  2495  2496  2497
                           2498  2499  249A  249B  249C  249D  249E  249F
                           R=(6.3us)  S=(1800)
1:00000:0002 G=(6.4us)     T=(270:15:14:00.917.220) DT=(  0:00:00:00.000.735)
    B MD                   C=1811( 3-R- 0-17)  Synch w/dat
                           0000
                           R=(6.8us)  S=(1800)
1:00000:0003 G=(2316.5us)T=(270:15:14:00.917.292) DT=(  0:00:00:00.000.072)
    A BR                   C=0820( 1-R- 1-32)
                           0001  0002  0003  0004  0005  0006  0007  0008
                           0009  000A  000B  000C  000D  000E  000F  0010
                           0011  0012  0013  0014  0015  0016  0017  0018
                           0019  001A  001B  001C  001D  001E  001F  0020
                           R=(6.3us)  S=(0800)
1:00000:0004 G=(696.5us) T=(270:15:14:00.917.980) DT=(  0:00:00:00.000.688)
    B BR                   C=0820( 1-R- 1-32)
```

Figure 3.5.1: Monitor Control Panel

The Monitor Control Panel consists of a menu bar and four boxes:

➢ The Menu Bar permits the user to configure or print the screen, save data, and define filter and trigger conditions.

- ➢ The Buffer View box displays the contents of the monitor buffer.
- ➢ The Display Control box permits the user to browse through the contents of the monitor buffer.
- ➢ The Monitor Control box allows the user to start and stop the monitoring of bus data.
- ➢ The Quick Look Monitoring box permits the user to update and view the monitor buffer at specified intervals.

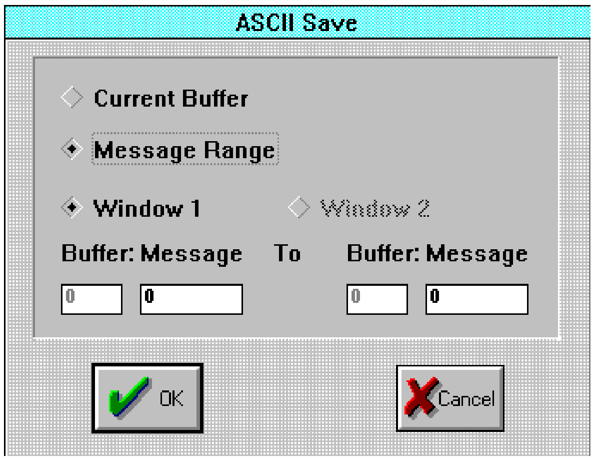
3.5.2 Menu Bar

The menu bar displays six menus that offer a list of commands or actions which may be executed to monitor, filter, print, save, and analyze data.

File Menu Use the File menu to save data from the monitor buffer to an ASCII text file or a binary archive file, print the data currently displayed on the screen, or exit from Snapshot Mode.

ASCII Save The ASCII Save Window (shown in Figure 3.5.2) allows the user to specify the entire current buffer or a selected message range to save as text to a designated file. This file is an ASCII text file formatted in the same manner as the PASS-1000 Monitor window which can be printed or edited using any ASCII text editor.

If two devices are being displayed in a split windows configuration, the user must select which of the two windows is to be saved.


ASCII Save

Current Buffer [✓ Message Range]

[✓ Window 1] Window 2

Buffer: Message 0 0 To Buffer: Message 0 0

Figure 3.5.2: ASCII Save Dialog

3.5.3 Saving the Current Buffer

If the **Current Buffer** button is selected, the entire current buffer is saved in ASCII format. When the **OK** button is pressed, a file dialog window will be opened. The path and filename are specified in this dialog. The file save can be canceled by clicking on the Cancel button in either the ASCII Save or File Windows.

Given the PASS-1000 buffer size of 55,168 words and an approximate 8 to 1 expansion function for ASCII conversion, saving the entire buffer will result in a file approximately 400,000 bytes long. If the disk becomes full before the selected save is completed, a warning message will be displayed and the save will be halted on the last line. The large size of the full buffer files may also present a problem to some simple ASCII editors such as the Windows NotePad. However, most word processors and the Norton series of editors for DOS and Windows can handle files of this size.

3.5.4 Saving a Range of Messages

If the **Message Range** button is selected, four text edit boxes will appear. These boxes are for specifying the range (first and last values) of messages and buffers to be saved. In the Snapshot mode, the buffer selections are preset at zero. In the Data logging mode, these fields can be set to any buffer number in the currently viewed file. When the **OK** Button is selected, a file window is opened to allow the user to specify the file name and path. The message range specified will then be copied to the designated file.

3.5.5 Archive Save

Choose the **Save as Archive File** option to save the current buffer to a binary file. Select this menu option, enter the desired filename in the File Open dialog box, and click **OK**. Cancel this option by clicking on the **Cancel** button.

3.5.6 Print Screen

Select this option to print the messages currently displayed in the Buffer View box on the default Windows printer.

3.5.7 Log Devices

When the PASS system includes multiple streams, choose this menu option to select which of the devices are to be monitored.

3.5.8 Log Filters

Select the desired device under the Log Filters menu to bring up the RT Filter Control Panel (shown in Figure 3.5.3). The RT Filter Control Panel allows the user to control what information is collected and stored by the monitor. This feature prevents the monitor buffer from filling with unnecessary data. A message is filtered based upon RT and subddress fields in the command word.

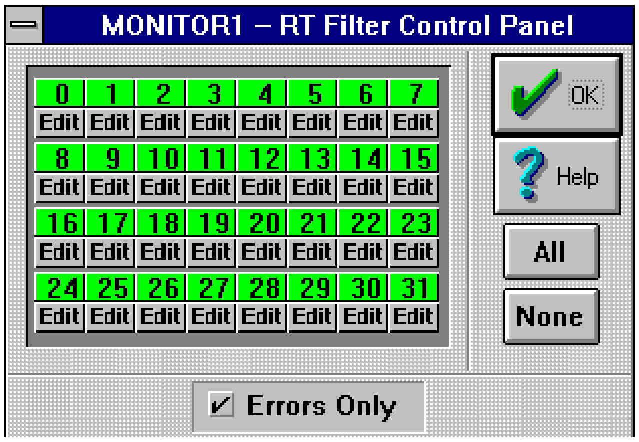


Figure 3.5.3: RT Filter Control Panel

This window contains a grid of 32 push-button pairs: one pair for each possible RT address. The upper button of each pair is used to select the RT and to indicate the current mode and status of the RT. The lower half of the button opens the next window level which allows subaddresses to be enabled. The All button will select all RTs. The None button will de-select all RTs. To select a specific RT, click the upper half of the desired RT push-button. The status of the RT is expressed with the colors listed in Table 3.5.1.

Table 3.5.1: RT Status Colors

Color	Description
Green	Indicates that ALL subaddresses of the RT have been selected.
Yellow	Indicates that SOME subaddresses of the RT have been selected.
Gray	Indicates that NONE of the RT’s subaddresses have been selected.

The **Errors Only** check box can be selected to monitor only errors being sent to the selected RT-SAs. The RT-SAs must still be selected using their respective dialogs for Error Only monitoring.

SA Filter Definition Panel

To select or de-select a subaddress, choose an Edit button from the RT Filter Control Panel. The SA Filter Definition Panel of the RT will appear (see Figure 3.5.4).

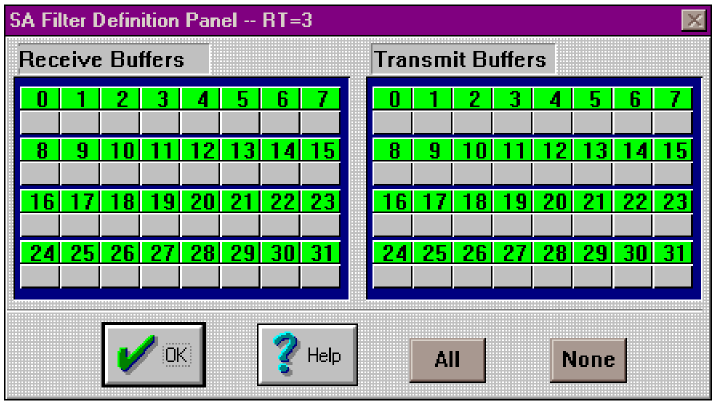


Figure 3.5.4: SA Filter Definition Panel

This window contains a grid of 64 buttons—one for each possible subaddress—that are used to select or deselect the associated subaddress. When a subaddress is selected, all messages directed at that subaddress will be stored in the monitor buffer. The **All** button will select all subaddresses of the RT. The **None** button will deselect all subaddresses. To select a specific subaddress, click the upper half of the desired subaddress button. The status of the subaddress is expressed with the colors listed in Table 3.5.2.

Table 3.5.2: Subaddress Status

Color	Description
Green	Indicates that the subaddress has been selected.
Gray	Indicates that the subaddress has *not* been selected.

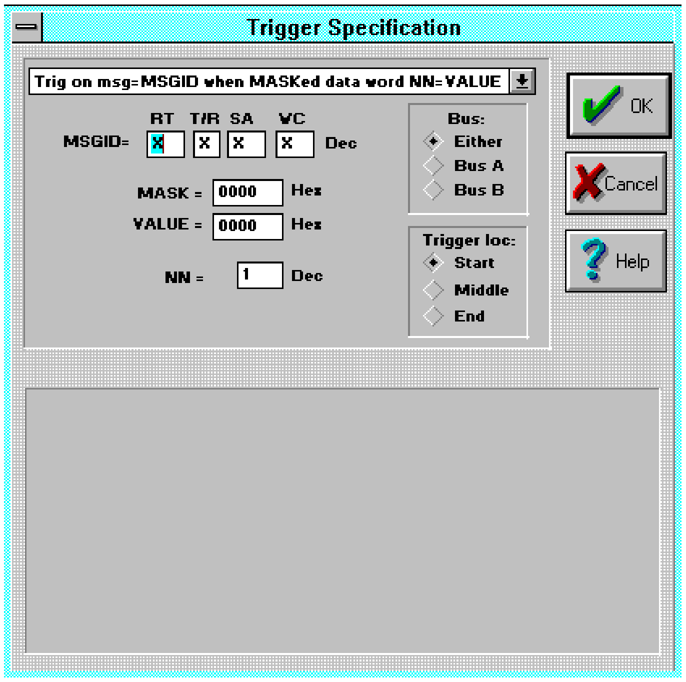
Triggers

It is sometimes useful to view bus activity around a defined point of interest. This point is called a trigger message and it controls when data is collected. The Trigger Specification window allows the user to define a trigger message by specifying a condition that must be satisfied by the status, command, and data words and message error activity. The monitor will stop and display the data when the trigger condition is satisfied and the monitor buffer is full.

Use the Trigger menu option to define the Trigger Specification for each device being monitored. There are two types of triggers which may be defined: Simple and Complex. The Trigger menu lists each device being monitored under each of the two trigger types. Select the desired device found under the desired trigger type to display the Trigger Specification window.

Every time the PASS triggers on data, it generates a pulse on pin 4 of the DB15 connector and can be used to trigger other measurement equipment, such as an oscilloscope. The signal on this cable is normally +5 volts. When triggered, this signal goes to zero volts for approximately 170 ns, then returns to +5 volts.

3.5.9 Defining a Simple Trigger



Trig on msg=MSGID when MASKed data word NN=VALUE

	RT	T/R	SA	WC	
MSGID=	X	X	X	X	Dec

MASK = 0000 Hex

VALUE = 0000 Hex

NN = 1 Dec

Bus: [✓ Either] Bus A Bus B

Trigger loc: [✓ Start] Middle End

Figure 3.5.5: Trigger Specification Window

The Trigger Specification window for defining a simple trigger (see Figure 3.5.5) has a drop-down box with ten possible trigger types and option buttons for bus selection and to define where to locate the trigger message in the monitor buffer. Fields named MSGID, MASK, VALUE, NN, and ERROR are associated with specific trigger types and will appear in the window when additional information is required.

Trigger Types To select one of the trigger types, click the down-arrow of the drop-down box to display a list of all trigger types, and then click the line of the desired trigger type. The trigger types in the drop-down box are described in Table 3.5.3.

Table 3.5.3: Trigger Types

Type	Description
Trigger on any msg	Triggers on any message that appears on the bus.
Trigger on msg= MSGID	Triggers on a message which has a command word equal to the MSGID field.
Trigger on msg=MSGID when MASKed data word NN=VALUE	Triggers on a message which has a command word equal to the MSGID field and a value in data word number NN equal to VAL-UE. The value of data word NN is logically ANDed with MASK and compared to VALUE. This comparison allows bit fields to be found within the data word.
Trigger on msg=MSGID when MASKed data word NN<>VALUE	Triggers on a message which has a command word equal to the MSGID field and a value in data word number NN NOT equal to VALUE. The value of data word NN is logically ANDed with MASK and compared to VALUE.
Trigger on msg=MSGID when MASKed status word = VALUE	Triggers on a message which has a command word equal to the MSGID field and a status word equal to VALUE. The status word is logically ANDed with MASK and compared to VALUE.
Trigger on msg=MSGID with MASKed status word <> VALUE	Triggers on a message which has a command word equal to the MSGID field and a status word NOT equal to VALUE. The sta-tus word is logically ANDed with MASK and compared to VALUE.
Trigger on any error	Triggers on a message which has an error.
Trigger on msg=MSGID with any error	Triggers on a message which has an error and a command word equal to the MSGID field.
Trigger on msg=MSGID with exact error combo= ERROR	Triggers on a message which has a command word equal to the MSGID field and an error code equal to ERROR (error code identical to selected errors).
Trigger on msg=MSGID with "and" of selected er-rors = ERROR	Triggers on a message which has a command word equal to the MSGID field and an error code which contains all selected errors and may also contain additional errors.
Trigger on msg=MSGID with "or" of selected errors = ERROR	Triggers on a message which has a command word equal to the MSGID field and an error code which contains at least one se-lected error.
Trigger on msgid=MSGID with error code <> ERROR	Triggers on a message which has a command word equal to the MSGID field and an error code NOT equal to ERROR.

Trigger Specifications

Five specifications allow the user to control and select which event must occur before the trigger condition is satisfied. They are associated with specific search types and will appear in the window as required. Change the value of the fields to create the trigger specification. Table 3.5.4 describes each field.

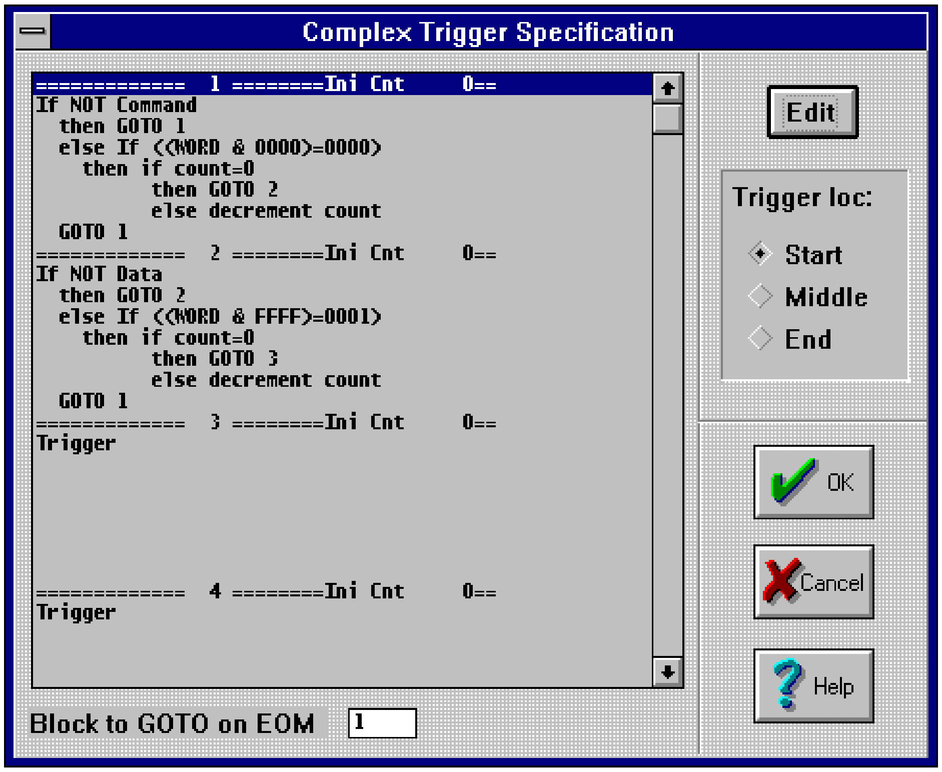
Table 3.5.4: Trigger Specification Fields

Field	Description
MSGID=	This field is the equivalent to the command word associated with a message. It consists of four subfields: ➢ RT: RT number; valid values are 0 through 31 and X. ➢ T/R: Transmit or Receive Bit; valid values are T, R, or X. ➢ SA: Subaddress number; valid values are 0 through 31 ➢ WC: Word count. When the MSGID field appears, it will contain don't care values or Xs. This value means that the field will be ignored and will not be used for comparison.
MASK=	This field is logically ANDed with the VALUE field to form a more complex trigger specification.
VALUE=	This field is used to define a value for a status or data word. It is logically ANDed with the MASK field.
NN=	This value specifies which data word in the message to trigger. This value must be between 1 and 32.
ERROR	Select the check box for the error to be found. There is a check box for each error that may be detected by the PASS-1000.
The Bus	Allows the user to define a trigger condition for only Bus A or Bus B. If the condition may apply to both buses, select Either.
The Trigger loc	Specifies where to locate the trigger message in the monitor buffer. It consists of the following option buttons: ➢ **Start:** Locate the trigger message at the beginning of the monitor buffer. ➢ **Middle:** Locate the trigger message in the middle of the monitor buffer. There must be enough messages to fill half of the buffer before the trigger message may be located in the middle. Otherwise, the trigger message will be located in the first half of the buffer. ➢ **End:** Locate the trigger message at the end of the monitor buffer. There must be enough messages to fill the buffer before the trigger message may be located at the end.

3.5.10 Defining a Complex Trigger

The Complex Trigger Specification allows the user to create and edit a customized trigger specification when the PASS-1000's pre-defined triggers are not satisfactory. This feature permits the internal structures of the PASS-1000, which are used for the trigger process, to be programmed. To display the Complex Trigger Specification window (shown in Figure 3.5.6), select the desired device under the "Complex" heading in the Triggers menu.

The internal structure consists of sixteen trigger control blocks that are executed according to the logic defined in each block. Only one trigger block is active at a time. The specification of the active trigger block is compared with the bus event. This comparison will initiate a trigger or a jump to another block.



```
=============  1 ========Ini Cnt     0==
If NOT Command
  then GOTO 1
  else If ((WORD & 0000)=0000)
    then if count=0
          then GOTO 2
          else decrement count
  GOTO 1
=============  2 ========Ini Cnt     0==
If NOT Data
  then GOTO 2
  else If ((WORD & FFFF)=0001)
    then if count=0
          then GOTO 3
          else decrement count
  GOTO 1
=============  3 ========Ini Cnt     0==
Trigger




=============  4 ========Ini Cnt     0==
Trigger
```

Trigger loc: [✓ Start] Middle End

Block to GOTO on EOM: 1

Figure 3.5.6: Complex Trigger Specification Window

The window consists of a view box that displays sixteen blocks and an **Edit** button. Use the vertical scroll bar to move through the view box. Clicking a line of a block highlights the line and makes that block active. The **Trigger loc:** box specifies where to locate the trigger message in the monitor buffer.

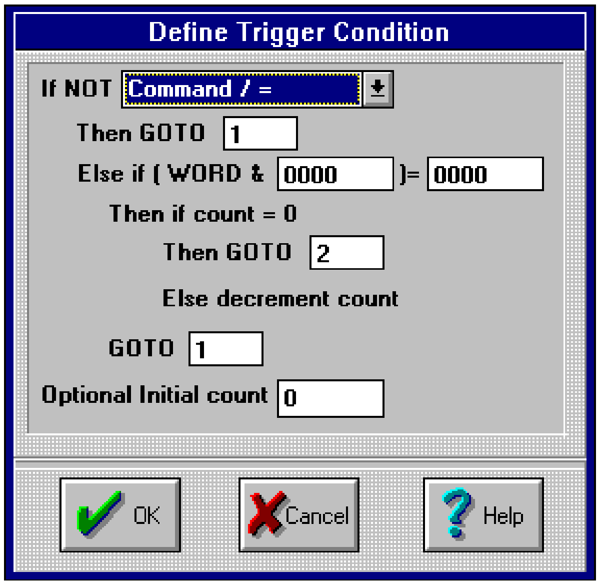
Cross Reference: The options are described in the "Defining a Simple Trigger" section.

The **Block to GOTO on EOM** text box, located at the bottom of the window, allows a jump to a block to be programmed at the detection of the End of Message (EOM). This feature is useful for defining trigger specifications for the second command or status words encountered in RT-RT messages and, also, prevents endless loops in the block structure.

Once the internal trigger structure has changed, it must be cleared before returning to the default trigger mode. If the user attempts to specify a simple trigger after a complex trigger has been defined, a warning window will appear with the message, “The existing trigger spec must be cleared to enter Simple trigger mode. Do you wish to continue?” Click **Yes** to clear the trigger specification and return the default trigger state.

Programming a Complex Trigger Condition

Click the Edit button to display the Define Trigger Condition window for the active block (see Figure 3.5.7).


Define Trigger Condition

If NOT Command / =

Then GOTO 1

Else if (WORD & 0000)= 0000

Then if count = 0

Then GOTO 2

Else decrement count

GOTO 1

Optional Initial count 0

Figure 3.5.7: Define Trigger Condition Window

The Define Trigger Condition window allows the user to define the contents of a specific block in the internal trigger structure. The window has a drop-down box where expression types are listed, followed by the rest of the block structure. The block structure has eight fields that may be programmed. They are used in the following format:

```
If NOT EXPRESSION
on BUS
Then GOTO GOTO1
Else if ( WORD & MASK )= VALUE
Then if count = 0
Then GOTO GOTO2
Else decrement count
GOTO GOTO3
Optional Initial count INITIAL COUNT
```

Each field is described in Table 3.5.5.

Table 3.5.5: Trigger Condition Fields

Field	Description
EXPRESSION	Defines a logical expression that the trigger block uses to compare with bus events. The expression consists of an event (the occurrence of a command, status, or data word or an error) and a comparison type (= equal, or <> not equal).
BUS	Specifies the bus on which the condition may occur.
GOTO1	Specifies the block to be executed when the first *if* statement is true.
MASK	Specifies the value of a mask that will be logically AND*ed* with the current word.
VALUE	Defines a value that will be compared to the result of the MASK logically AND*ed* with the current word value.
GOTO2	Specifies the block to be executed when the second *if* statement is true.
GOTO3	Specifies the block to be executed when the second *if* statement is false.
INITIAL COUNT	Defines the number of times to execute the block before jumping to the next block.

Expression Types

To select one of the expression types, click the down-arrow of the drop-down box to display a list of all expression types and then click the line of the desired expression type. The expression types in the drop-down box are described in Table 3.5.6.

Table 3.5.6: Expression Types

Type	Description
Trigger	Initiates the trigger. The trigger will be located in the monitor buffer according to the option selected in the **Trig loc:** box.
Command/=	Searches for a message with a command word equal to VALUE.
Command/<>	Searches for a message with a command word NOT equal to VALUE.
Status/=	Searches for a message with a status word equal to VALUE.
Status/<>	Searches for a message with a status word NOT equal to VALUE.
Data/=	Searches for a message with a data word equal to VALUE.
Data/<>	Searches for a message with a data word NOT equal to VALUE.
Error/=	Searches for a message with an error code equal to VALUE. See Appendix A for error code values appropriate for VALUE.
Error/<>	Searches for a message with an error code NOT equal to VALUE. See Appendix A for error code values appropriate for VALUE.

Trigger Block Execution

A trigger block follows a logic sequence of operations to determine if an event has occurred. It first compares bus activity with the current word or action specified in EXPRESSION. If the comparison fails, execution jumps to the block defined in GOTO1. If the comparison is true, the value of the current word is logically *ANDed* with MASK and compared to VALUE. If the result of this comparison is true and the INITIAL COUNT is zero, execution passes to the block defined in GOTO2. Otherwise, INITIAL COUNT is decremented and execution passes to the block defined in GOTO3.

Complex Trigger Examples

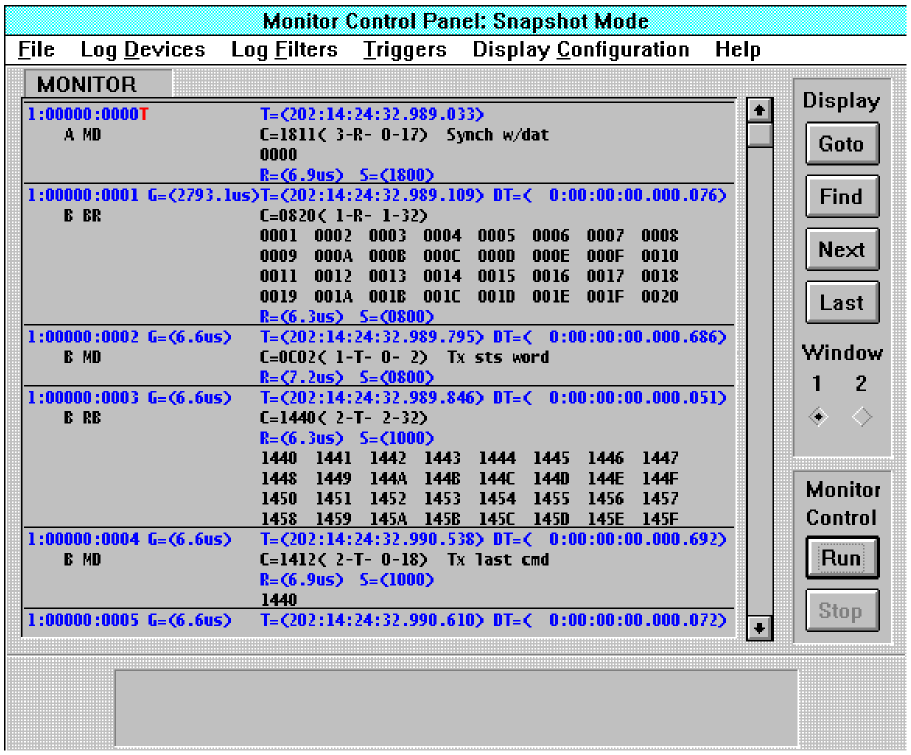
Complex triggering is best illustrated with examples. Simple triggers, those defined from the Trigger Specification window, are themselves a series of trigger blocks. Understanding how simple triggers are expressed as complex triggers is the first step to learning to program more sophisticated trigger specifications. To display a simple trigger in the form of trigger blocks, define a simple trigger specification from the Trigger Specification window. Display the Complex Trigger Specification window to view the simple trigger specification as one or more trigger blocks.

Below are two examples of complex trigger specifications. In the first example, the specification will trigger upon the third occurrence of a command word with RT=3, Receive, and SA=3. INITIAL COUNT must be set to two in order for the block to trigger after three occurrences.

```
=============  1 ========Ini Cnt     2==
If NOT Command
then GOTO 1
else If ( (WORD & FFE0 )= 1860)
then if count=0
then GOTO 2
else decrement count
GOTO 1
=============  2 ========Ini Cnt     0==
Trigger
This example triggers upon the third command word with RT=3, Receive, and
SA=3 that occurs after a Synchronize mode code to RT number 4.
=============  1 ========Ini Cnt     0==
If NOT Command
then GOTO 1
else If ( (WORD & FBFF )= 2001)
then if count=0
then GOTO 2
else decrement count
GOTO 1
=============  2 ========Ini Cnt     2==
If NOT Command
then GOTO 2
else If ( (WORD & FFE0 )= 1860)
then if count=0
then GOTO 3
else decrement count
GOTO 2
=============  3 ========Ini Cnt     0==
Trigger
```

Display Configuration

The monitor screen may be viewed as a single window or in a split screen configuration, as shown in Figure 3.5.8. To view as a split screen, select Setup/Split Window from the Display Configuration menu. The split windows may be configured to display different devices or the same device. Choose a device to be displayed in a window using the Devices option in the Display Configuration menu. The Delta Time box in the center of the split screen gives the time difference between the first message displayed in the top window versus that in the bottom. (This time will appear as a negative value if the first message in the bottom window occurred prior to that in the top.)



```
1:00000:0000T              T=(202:14:24:32.989.033)
     A MD                  C=1811( 3-R- 0-17)  Synch w/dat
                           0000
                           R=(6.9us)  S=(1800)
1:00000:0001 G=(2793.1us)T=(202:14:24:32.989.109) DT=(  0:00:00:00.000.076)
     B BR                  C=0820( 1-R- 1-32)
                           0001  0002  0003  0004  0005  0006  0007  0008
                           0009  000A  000B  000C  000D  000E  000F  0010
                           0011  0012  0013  0014  0015  0016  0017  0018
                           0019  001A  001B  001C  001D  001E  001F  0020
                           R=(6.3us)  S=(0800)
1:00000:0002 G=(6.6us)     T=(202:14:24:32.989.795) DT=(  0:00:00:00.000.686)
     B MD                  C=0C02( 1-T- 0- 2)  Tx sts word
                           R=(7.2us)  S=(0800)
1:00000:0003 G=(6.6us)     T=(202:14:24:32.989.846) DT=(  0:00:00:00.000.051)
     B RB                  C=1440( 2-T- 2-32)
                           R=(6.3us)  S=(1000)
                           1440  1441  1442  1443  1444  1445  1446  1447
                           1448  1449  144A  144B  144C  144D  144E  144F
                           1450  1451  1452  1453  1454  1455  1456  1457
                           1458  1459  145A  145B  145C  145D  145E  145F
1:00000:0004 G=(6.6us)     T=(202:14:24:32.990.538) DT=(  0:00:00:00.000.692)
     B MD                  C=1412( 2-T- 0-18)  Tx last cmd
                           R=(6.9us)  S=(1000)
                           1440
1:00000:0005 G=(6.6us)     T=(202:14:24:32.990.610) DT=(  0:00:00:00.000.072)
```

Figure 3.5.8: Monitor Control Panel: Snapshot Mode

3.5.11 Buffer View Box

The Buffer View box displays MIL-STD-1553 messages and spurious data captured by the monitor buffer. A message consists of a blue timing line, a command line, data, a status response line, and an error message(s) if necessary.

Timing Four time values are recorded for a message by the PASS-1000: the absolute time (T), the delta time (DT), the response time (R), and the gap time (G). The first line of message information is a blue timing line. It displays the stream, buffer, and message number followed by the gap, absolute, and delta times. The first message captured by the monitor will not display the delta time. The times are defined in Table 3.5.7.

Table 3.5.7: Time Value Recorded for a Message

Value	Description
T (Absolute Time)	The value of the internal ABI-PASS timer when the message was received. This timer is set from the PC clock each time the PASS-1000 program is started. The format is: days : hours : minutes : seconds . milliseconds . microseconds
DT (Delta Time)	The difference between the starting time of the previous message and the starting time of the current message. The format is: days : hours : minutes : seconds . milliseconds . microseconds
G (Gap Time)	The number of microseconds between the previous message and the current message. It has a resolution of 1/10th of a microsecond. If the gap time is greater than six milliseconds, the string >6*ms* will be displayed.
R (Response Time)	The number of microseconds it takes for the RT to respond to a valid command.

The previous message is defined as the message transmitted *on the same bus* as the current message and received immediately before this message. It is not necessarily the previous message received and written to the monitor buffer.

The time information at the beginning of each message is usually displayed in blue. If an external IRIG signal is used as the time source, the color of the time information indicates whether or not the IRIG signal is present. The time information is displayed in blue if the IRIG signal is present and displayed in red if the signal is not present.

Command Line This line follows the timing line and contains bus information, the message type, and the command word for the message. It has the following format: bus type command (see Table 3.5.8 for more details).

Table 3.5.8: Command Line Format

Flag	Description
bus	This flag indicates the bus over which the message was received. It may have one of the following values: ➢ A: Primary bus ➢ B: Secondary bus
type	The next flag describes the type of transfer. It may have one or more of the following values: ➢ BR: Bus Controller to Remote Terminal ➢ RB: Remote Terminal to Bus Controller ➢ RR: Remote Terminal to Remote Terminal ➢ MD: Mode Command
RT label	If the RT has been assigned a name using the RT Labels dialog in the Tool menu, this label will be displayed next.
command	The command word is represented by ***C*=**value(RT#-T/R-SA#-WC/MC), where value is the command word in hexadecimal, RT# and SA# are the remote terminal and subaddress numbers, T/R is Transmit or Receive, and WC/MC is the word count or mode code. If the message is a mode command, a short description of the mode code will follow the command word.

Data and Response Lines The data words and the status response are displayed on the following lines in the order in which they were transmitted. The response line contains the response time which is represented by **R=***(microseconds)* and the status word which is displayed as **S=***(value)*. Data words are displayed in hexadecimal, with eight data words per line.

Trigger Flag This flag (**T**) is displayed in red to the left of the absolute time, a command word (**C**), a response line (**R**), or a data word. It identifies the trigger message; i.e., the message which satisfied the trigger condition. In some cases, the trigger flag will be displayed on the message line following the trigger message.

Error Messages If the message has an error, a message or character describing the type of error and its location will be written in red. See Appendix A for further details regarding error messages.

Scanning the Monitor Buffer Use the vertical scroll bar to browse through the monitor buffer. Clicking either scroll arrow will move the display down or up one message. Clicking the bar below the scroll box will move the display forward ten messages. Clicking above the scroll box will move it back ten messages. The focus of the Buffer View box may also be set to map the PAGE UP, PAGE DOWN, UP ARROW, DOWN ARROW, HOME, and END keys. Move the mouse cursor over the Buffer View box and click the left button. A box with a dotted line will appear outlining the messages in the Buffer View box. Now use the PAGE UP, PAGE DOWN, UP AR-

ROW, and DOWN ARROW keys to move forwards or backwards, the HOME key to return to the top of the buffer, and the END key to move to the end of the buffer.

3.5.12 Display Control Box

The Display Control box allows the user to browse through the monitor buffer. It contains four buttons and a window select option. The window select option designates which window will be affected by the button commands when using a split screen configuration. The button functions are described in Table 3.5.9.

Table 3.5.9: Display Control Box Buttons

Button	Description
Goto	Pops up the GOTO Message window which allows the user to view a specific message in the buffer.
Find	Pops up the Find Specification window which permits the user to define a search pattern. The first message that meets the search condition is displayed at the top of the Buffer View box. If none of the messages meet the search condition, an error window will appear with the message *Message not found!*
Next	Searches for the next message that meets the search pattern and displays it at the top of the Buffer View box. If none of the following messages meet the search condition, an error window will appear with the message *Message not found!*
Last	Searches for the last message that meets the search pattern and displays it in the Buffer View box. If none of the following messages meet the search condition, an error window will appear with the message *Message not found!*

GOTO Message Box

The GOTO Message window (shown in Figure 3.5.9) allows the user to specify a particular message to view in the Buffer View box. Click the GOTO button in the Display control box to display this window.

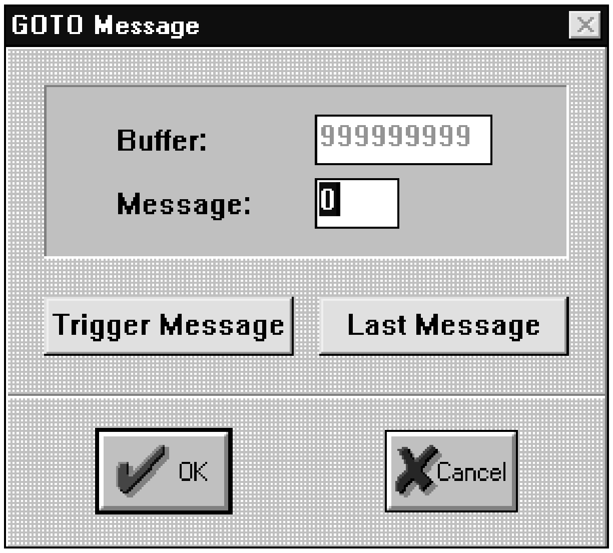
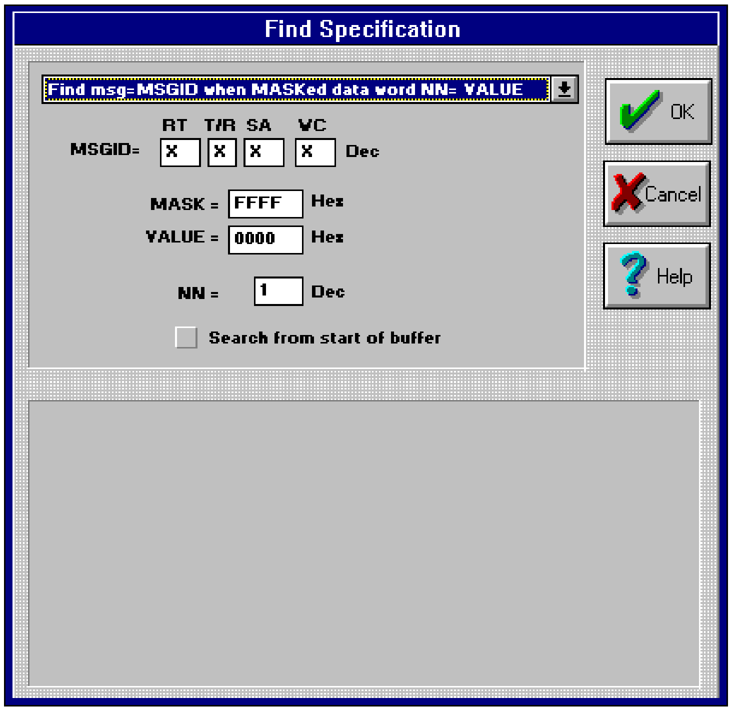


Figure 3.5.9: GOTO Message Window

To view a specific message, enter the desired message number in the **Message:** option box and click **OK**. Click the Trigger Message button to display the trigger message at the top of the Buffer View box. If the trigger message is not in the monitor buffer, an error window will appear with the message *Monitor Buffer did not include specified Trigger!* Click the Last Message button to display the last two messages in the Buffer View box. The **Buffer:** option box is only active in Data Logging mode.

Finding Messages

The Find Specification window (shown in Figure 3.5.10) allows the user to define the search condition with which to locate messages in the monitor buffer. This feature is particularly useful when working with large collections of assorted data. Search for message errors, distinct command and status words, and values within data words. Click the **Find** button in the Monitor Control Panel to display the Find Specification window.



Find msg=MSGID when MASKed data word NN= VALUE

	RT	T/R	SA	WC	
MSGID=	X	X	X	X	Dec

MASK = FFFF Hex
VALUE = 0000 Hex
NN = 1 Dec
Search from start of buffer

Figure 3.5.10: Find Specification Window

This window has a drop-down box with ten possible search types and a check box to specify the location in the monitor buffer where the search is to begin. Fields named MSGID, MASK, VALUE, NN, and ERROR are associated with specific search types and will appear in the window when additional information is required.

Search Types To select one of the search types, click the down-arrow of the drop-down box. A list of search types will appear. Click the line of the desired search type. The search types in the drop-down box are described in Table 3.5.10.

Tip: A search by time stamp may be used to locate a specific block of messages in an archive file which may then be deleted, creating a new archive file for playback.

Table 3.5.10: Search Types

Type	Description
Find msg=MSGID	Searches for a message which has a command word equal to the MSGID field.
Find msg<>MSGID	Searches for a message which has a command word NOT equal to the MSGID field.
Find msg=MSGID when MASKed data word NN=VALUE	Searches for a message which has a command word equal to the MSGID field and a value in data word number NN equal to VALUE. The value of data word NN is logically *ANDed* with MASK and compared to VALUE. This comparison allows bit fields to be found within the data word.
Find msg=MSGID when MASKed data word NN<>VALUE	Searches for a message which has a command word equal to the MSGID field and a value in data word number NN NOT equal to VALUE. The value of data word NN is logically *ANDed* with MASK and compared to VALUE.
Find msg=MSGID with MASKed status word = VALUE	Searches for a message which has a command word equal to the MSGID field and a status word equal to VALUE. The status word value is logically *ANDed* with MASK and compared to VALUE.
Find msg=MSGID with MASKed status word <> VALUE	Searches for a message which has a command word equal to the MSGID field and a status word NOT equal to VALUE. The status word value is logically *ANDed* with MASK and compared to VALUE.
Find any error	Searches for a message which has an error.
Find msg=MSGID with any error	Searches for a message which has an error and a command word equal to the MSGID field.
Find msgid=MSGID with error code = ERROR	Searches for a message which has a command word equal to the MSGID field and an error code equal to ERROR.
Find msgid=MSGID with error code <> ERROR	Searches for a message which has a command word equal to the MSGID field and an error code NOT equal to ERROR.
Find msg with Time >= Day:Hr:Min:Sec:Msec:Usec	Searches for a message which has a time stamp greater than or equal to Day:Hr:Min:Sec:Msec:Usec.
Find msg with Delta Time >= Day:Hr:Min:Sec:Msec:Usec	Searches for a message which has a delta time value greater than or equal to Day:Hr:Min:Sec:Msec:Usec.
Find msg with Delta Time <= Day:Hr:Min:Sec:Msec:Usec	Searches for a message which has a delta time value less than or equal to Day:Hr:Min:Sec:Msec:Usec.
Find msg with Gap Time >= Usec	Searches for a message which has a gap time greater than or equal to Usec.

Type	Description
Find msg with Gap Time <= Usec	Searches for a message which has a gap time greater than or equal to Usec.

Search Specifications Five specifications allow the user to control and select which event must occur before the search condition is satisfied. They are associated with specific search types and will appear in the window as required. Change the value of the fields to create the desired find specification. Table 3.5.11 contains a description of each field.

Table 3.5.11: Search Specifications

Field	Description
MSGID=	This field is the equivalent to the command word associated with a message. It consists of four subfields: ➢ RT: RT number; valid values are 0 through 31 and X. ➢ T/R: Transmit or Receive Bit; valid values are T, R, or X. ➢ SA: Subaddress number; valid values are 0 through 31 or X. ➢ WC: Word count; valid values are 1 through 32 or X. When the MSGID field appears, it will contain *don't care* values or Xs. This value means that the field will be ignored and will not be used for comparison.
MASK=	This field is logically *ANDed* with the VALUE field to form a more complex find specification.
VALUE=	This field is used to define a value for a status or data word. It is logically *ANDed* with the MASK field.
NN=	This value specifies which data word to find in the message. This value must be between 1 and 32.
ERROR	Select the check box for the error to be found. There is a check box for each error that may be detected by the PASS-1000. Multiple selections result in a search specification that will attempt to find the occurrence of either error; in other words, the errors are combined in an *OR* function.

If the **Search from start of buffer** check box is checked, the search will start from the beginning of the monitor buffer. An empty check box will start the search from the current message.

3.5.13 Monitor Control Box

The Monitor Control Box allows the user to start and stop monitoring bus data. It consists of the two buttons described in .

Table 3.5.12: Monitor Control Box Buttons

Button	Description
Run	Enables monitoring operations. Data is stored in the monitor buffer after the trigger condition is satisfied. When the buffer is full, the contents of the buffer will be displayed in the Buffer View box.
Stop	Disables the monitor and displays the contents of the monitor buffer.

3.5.14 Quick Look Monitoring

The Quick Look Monitoring box (shown in Figure 3.5.11) contains controls which allow the user to specify intervals at which to update and view the monitor buffer.

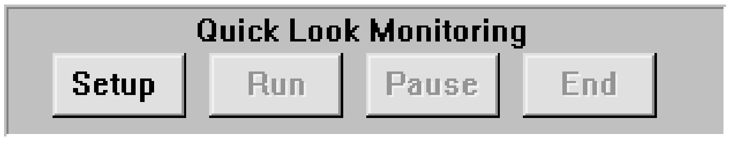


Figure 3.5.11: Quick Look Monitoring Box

Click **Setup** to open the Quick Look Monitor Setup window (shown in Figure 3.5.12).

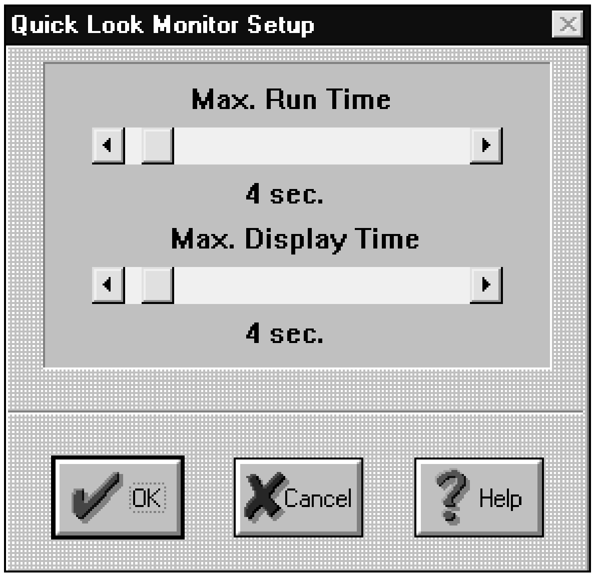


Figure 3.5.12: Quick Look Monitor Setup Window

The Quick Look Monitor Setup window contains two horizontal scroll bars: Max. Run Time and Max. Display Time. Max. Run Time determines the length of time in which the monitor buffer is filled with data. If the buffer fills before the specified time expires, the data will be displayed. Max. Display Time determines the length of time in which the buffer is displayed in the Monitor window. The default value for each of these time intervals is four seconds. Click on the right or left arrows to adjust these intervals to the desired values. Click **OK** to save the changes and return to the Monitor window. Click **Cancel** to abandon the changes.

Click the **Run** button in the Quick Look Monitoring box to begin updating the monitor buffer. Suspend Quick Look Monitoring at any time by clicking the **Pause** button. Return to normal Snapshot mode by clicking the **End** button.

3.6 Data Logging

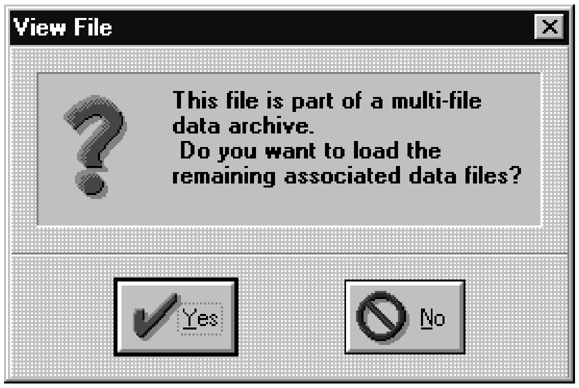
The Data Logging mode monitors traffic on the MIL-STD-1553 bus and stores the information onto a device such as a hard disk or tape. The data may be stored onto multiple disk partitions across multiple drives.

The features of Data Logging are similar to that of the Snapshot mode of the Bus Monitor in that the user can monitor data, filter messages, trigger on specific messages, examine the buffer, and store it to a device. In Data Logging mode, the data may be recalled by the PASS-1000 for viewing or processing or by a custom application. It may also be reproduced using the Playback feature, described in a section found later in this manual.

3.6.1 Monitor Control Panel: Data Logging

From the Monitor menu, choose Data Logging Mode. The File Open window will appear, requesting a file name. Type a filename to create a new file or choose a file from the Files list. By default, archive files have an extension of .ARC.

If an existing log file that is part of a multiple partition archive is chosen, the dialog shown in Figure 3.6.1 appears.


View File

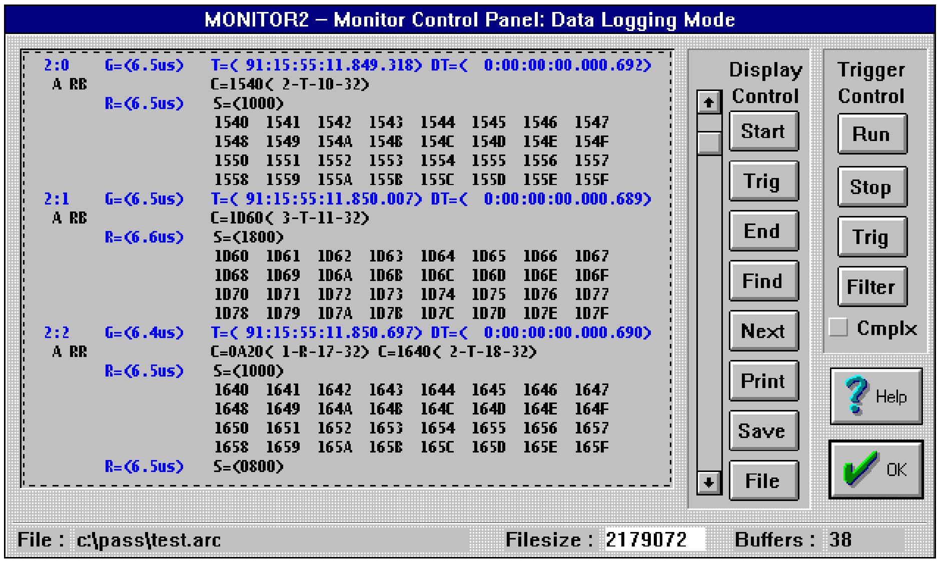
This file is part of a multi-file data archive.
Do you want to load the remaining associated data files?

Figure 3.6.1: View File Dialog

Select **No** to view the initial log file only; select **Yes** to view all of the data that was archived with this log file.

Cross Reference: See the Archived Data description which follows in this section for more information about multiple partition archives.

After a filename has been accepted, the Monitor Control Panel for Data Logging (shown in Figure 3.6.2) will appear. If an existing log file is chosen, the displayed RT Labels will be those assigned when the file was created.



```
  2:0     G=(6.5us)     T=( 91:15:55:11.849.318) DT=(  0:00:00:00.000.692)
   A RB                 C=1540( 2-T-10-32)
          R=(6.5us)      S=(1000)
                         1540  1541  1542  1543  1544  1545  1546  1547
                         1548  1549  154A  154B  154C  154D  154E  154F
                         1550  1551  1552  1553  1554  1555  1556  1557
                         1558  1559  155A  155B  155C  155D  155E  155F
  2:1     G=(6.5us)     T=( 91:15:55:11.850.007) DT=(  0:00:00:00.000.689)
   A RB                 C=1D60( 3-T-11-32)
          R=(6.6us)      S=(1800)
                         1D60  1D61  1D62  1D63  1D64  1D65  1D66  1D67
                         1D68  1D69  1D6A  1D6B  1D6C  1D6D  1D6E  1D6F
                         1D70  1D71  1D72  1D73  1D74  1D75  1D76  1D77
                         1D78  1D79  1D7A  1D7B  1D7C  1D7D  1D7E  1D7F
  2:2     G=(6.4us)     T=( 91:15:55:11.850.697) DT=(  0:00:00:00.000.690)
   A RR                 C=0A20( 1-R-17-32) C=1640( 2-T-18-32)
          R=(6.5us)      S=(1000)
                         1640  1641  1642  1643  1644  1645  1646  1647
                         1648  1649  164A  164B  164C  164D  164E  164F
                         1650  1651  1652  1653  1654  1655  1656  1657
                         1658  1659  165A  165B  165C  165D  165E  165F
          R=(6.5us)      S=(0800)
```

File : c:\pass\test.arc   Filesize : 2179072   Buffers : 38

Figure 3.6.2: Data Logging Control Panel

The Data Logging Control Panel is identical to the Monitor Control Panel: Snapshot Mode except for the addition of a Data Logging Status Bar and the ability to scroll by buffers and files in addition to messages. The menu options and the Display and Monitor Control buttons have the same function as those found in Snapshot Mode, with the exception of the Load New File, Binary Save, and Archive Disk Usage options, described below.

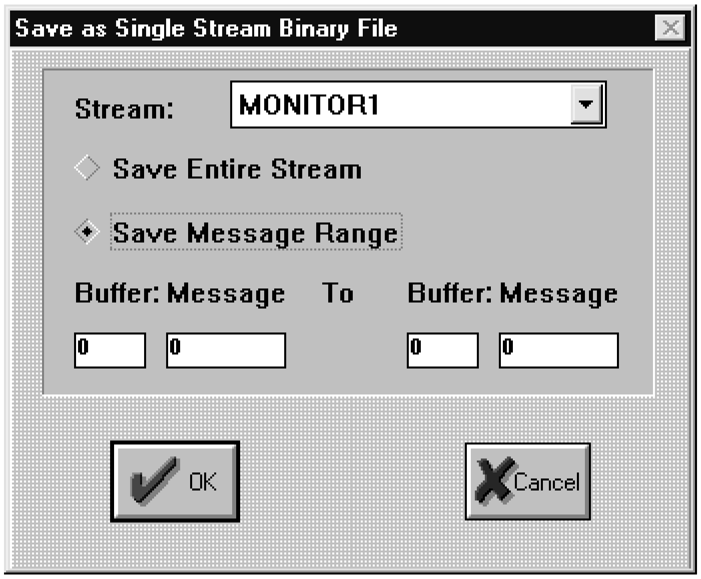
Cross Reference: Refer to the “Bus Monitoring” section for more information on all other functions.

Warning: Do not attempt to perform any other operation during an Archive session, including resizing a window, or switching to another windows application. Any such operation will adversely affect the stored data.

Load New File Choose the **Load New File** option in the Data Logging File menu to view a different archive file or create a new file.

Binary Save

Choose the **Save as Binary File** option in the Data Logging File menu to save the selected stream to a binary file. The binary file will contain bus activity from a single stream in the currently viewed file only.



Save as Single Stream Binary File

Stream: MONITOR1

Save Entire Stream [✓ Save Message Range]

Buffer: Message		To	Buffer: Message	
0	0		0	0

Figure 3.6.3: Save as Single Stream Binary File Window

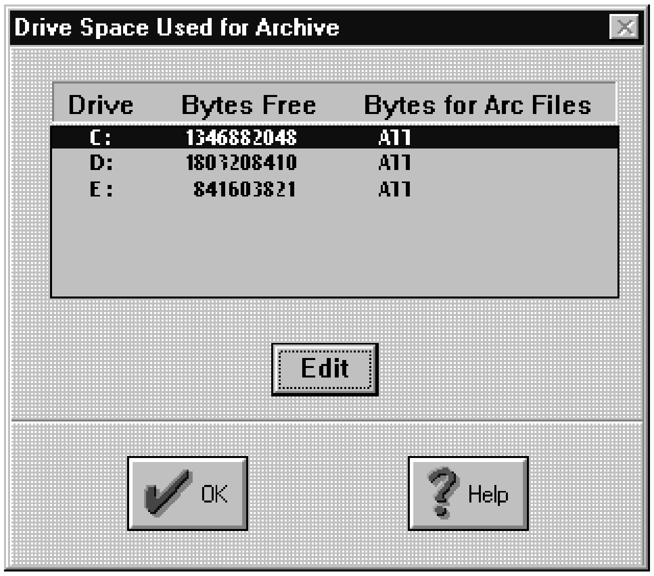
Select the desired stream from the drop-down box in the Save as Single Stream Binary File window (see Figure 3.6.3). Click on **Save Entire Stream** to save all bus activity for the device to the binary file. Click on **Save Message Range** to save only a selected message range to the file. If this option is chosen, four text edit boxes appear. Enter the first and last Buffer: Message values for the range to be saved. These fields can be set to any buffer number in the currently viewed file. Click **OK** to open a file window and specify the path and file name. Click **OK** in the File Open window to create the binary file.

Tip: Use the Binary Save option in conjunction with the Find button to save blocks of messages for playback.

Archive Disk Usage

Archive data may be stored on multiple partitions across multiple disk drives. (These drives must be hard drives local to the host computer, not network drives.) The PASS automatically detects the available drives and the number of bytes free on each. Choose the **Archive Disk Usage** option in the Data Logging File menu to specify the number of bytes on each drive to be used for archive files.

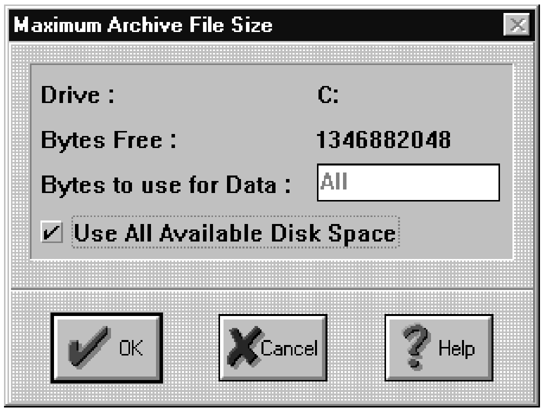
Note: The PASS software does not detect the size of removable or network drives. They are displayed in the Drive Space Used for Archive (see Figure 3.6.4) as NETWORK with no value listed under Bytes Free. To specify the Bytes for Arc Files, first verify that the drive is accessible. Then, double-click on the desired drive or highlight the drive and click **Edit**. A Warning box will appear, stating that this is a removable or network drive and that it must be accessible in order to continue. If the drive is accessible, click **OK** to display the Bytes Free and to specify the Bytes for Arc Files value. **If the drive is not accessible, an error will be generated and the system may crash.**



Drive	Bytes Free	Bytes for Arc Files
C:	1346882048	All
D:	1803208410	All
E:	841603821	All

Figure 3.6.4: Drive Space Dialog

The default number of bytes is All for each detected drive. To change this value, highlight the desired drive and click the **Edit** button.



Drive : C:

Bytes Free : 1346882048

Bytes to use for Data : All

☑ Use All Available Disk Space

Figure 3.6.5: Maximum Archive File Size Dialog

The dialog box shown in Figure 3.6.5 appears. Specify the desired number of Bytes to use for Data in the text box provided and click OK to effect the change.

Note: If the archive Filesize specified in the Data Logging Status Bar exceeds the total number of bytes allocated for archive files, a message box will appear and archiving will not occur.

Data is archived as described in Table 3.6.1.

Table 3.6.1: Data Archive Files

Archive File	Description
1st Archive File	When an archive session begins, the PASS stores the first archive file in the \PASS\DBASE directory on the current partition; i.e., the partition on which the PASS software is resident. The extension for the first archive file is .ARC.
2nd Archive File	If the data to be archived exceeds the file size allocated on the current partition, a second archive file is created. This file has the same filename as the first archive file, but with a .1 file extension. The second archive file is stored in the \PASSDATA directory on the next partition to have space allocated for an archive operation.
Subsequent Archive Files	If subsequent archive files are required, they are assigned file extensions of .2, .3, and so forth. Each of these files is stored in the *\passdata* directory on the next partition to have space allocated for an archive operation.

Note: The PASS software creates the \PASSDATA directory, when required, during an archive session.

Table 3.6.2 and Table 3.6.3 illustrate archive filenames and storage locations when multiple partitions are being used. In the example configuration, the PASS software is resident on D.

Table 3.6.2: Archive Filenames

Drive	Bytes Used for Archive	Directories used for Archive Files
D	1G	D:\PASS\DBASE
E	1.8G	E:\PASSDATA
F	1.8G	F:\PASSDATA
C	1.8G	C:\PASSDATA

For an archive operation that stores 6G bytes of data, four archive files are required. Based on the configuration shown in Table 3.6.2, storage will occur as shown in Table 3.6.3.

Table 3.6.3: Archive Storage Locations

Archive Filename	Storage Path
EXAMPLE.ARC	D:\PASS\DBASE
EXAMPLE.1	E:\PASSDATA
EXAMPLE.2	F:\PASSDATA
EXAMPLE.3	C:\PASSDATA

Data Logging Status Bar

The Data Logging Status Bar located at the bottom of the Monitor Control Panel displays file information such as a file pathname, the file size, the number of files, the number of buffers in the files, and the number of device streams archived in the files. The Filesize text box specifies the total number of bytes to be stored during the archive session. This value can be as large as all of the available disk space on all of the available partitions. Edit this text box to reserve more or less space.

The maximum size of each archive file is about two megabytes; therefore, the value in the Files: field is determined by the Filesize value. For example, for an archive Filesize of ten megabytes, five files would be required.

3.6.2 Archived Data

Data is archived as a collection of buffers. One buffer is equal to the maximum amount of data that can be captured and stored by the PASS-1000 without any data overrun. A buffer is equal to 55,160 bytes in the current software and firmware versions. PASS-1000 Versions 1.40 to 1.47 use a buffer size of 57,216 bytes while versions earlier than 1.40 use a buffer size of 57,344 bytes.

In the Buffer View box, messages are designated on the timing line by their stream, file, buffer, and message number. For example, a message number of 1:2:3:661 indicates the 661th message in buffer number three of file number two of stream one. File and buffers begin at zero and streams begin at one.

Scanning the Data Logging Buffer

The archive buffer may be browsed in increments of messages, buffers, or files. From one to three vertical scroll bars will be present, depending on the number of buffers and files.

Messages The inner-most scroll bar (nearest to the Buffer View Box) moves the display in relation to the messages.

Buffers If the archive information contains more than one buffer, a second scroll bar will be present to the right of the message scroll bar. This scroll bar moves the display in relation to the buffers.

Files If the archive information contains more than one file, a third scroll bar will be present in the outer-most position. This scroll bar moves the display in relation to the files.

The following instructions apply to each of the three scroll bars: Click either scroll arrow on the scroll bar to move the display down or up one message (or buffer or file). Click the bar below the scroll box to move the display forward ten messages (or buffers or files). Click above the scroll box to move it back ten messages (or buffers or files).

The focus of the Buffer View box may be set to map the PAGE UP, PAGE DOWN, UP ARROW, DOWN ARROW, HOME, and END keys. Move the mouse cursor over the Buffer View box and click the left button. A box with a dotted line will appear outlining the messages in the Buffer View box. Now use the PAGE UP, PAGE DOWN, UP ARROW, and DOWN ARROW keys to move forwards or backwards, the Home key to return to the top of the buffer, and the End key to move to the end of the buffer.

3.6.3 Data Logging File Format

Two distinct types of data are stored in each archive file: header and footer information used by the PASS-1000 software to recreate the data display and the actual binary data received from the bus. The binary data is stored in 55,168 byte blocks, corresponding to the memory buffers used by the ABI-PASS board. Each of these buffers is written to the file as 431 128-byte records to maximize the efficiency of the disk write. In order to match the five word data structure used by the ABI-PASS in classifying bus data and the 128-byte record used in the disk write operation, an additional four words (eight bytes) is added to the end of the buffer and is discarded on playback.

The header information is also written to a 55,168 byte block. This allows further expansion and simplifies outside processing of the binary bus data. The header is contained in the first 128-byte record and consists of the fields described in Table 3.6.4.

Table 3.6.4: Header Format

Field	Type	Length
Version Flag	array[0..4] of char	5 bytes
Version Number	Type Word	2 bytes
Buffer Size	Type Long Integer	4 bytes
Number of Records	Type Word	2 bytes
Number of Streams	Type Word	2 bytes
IRIG Card Used	array[1..MaxStream] of Boolean	8 bytes
Stream Names	array[1..MaxStream,0..10] of char	88 bytes
Stream Type	array[1..MaxStream] of word	16 bytes
Total Files in Archive	1 byte	1 byte

The **version flag** is a character string flag used to distinguish this header structure from previous header versions. The character flag is used to facilitate interpretation of the header by other programming languages, especially C or C++. The **version number** is used by the PASS-1000 software to recognize older file formats for display and is now stored as a word. Previous versions of the PASS-1000 used a real number file version which proved to be incompatible with other languages. The **buffer size** field gives the size of the buffer used in the file in bytes while the **number of records** field gives the size of the buffer in 128-byte records. The **number of streams** variable lists the number of ABI-PASS devices that actively wrote data to the file. The array of **IRIG** booleans indicate whether an IRIG card was used to provide an external time signal for each active device when the data logging file was created. This allows the PASS-1000 software to display message times in red for messages that arrived when the IRIG card was not synchronized with the external time source. The **stream names** fields are used to store names for each active device. MaxStream is the maximum number of ABI-PASS devices supported by the software and currently is set to eight devices. The **stream type** field specifies the type of PASS card: 1553 PC2, 1553 PC3 (also used for 1553 PCMCIA), 429, or 194. The **total files** field specifies the number of files opened at the time of the archive session.

The next three 128-byte records of the header block contain a table for RT-labels. The next four 128-byte records contain an additional table for designating 1553A RTs and user defined 1553A mode codes. The next 80 characters contain the name of the next file in the archive. The rest of the 55,168 byte header block is left for future expansion.

A single 128-byte footer is also written to the file at the end of the data buffers. This footer contains an array of long (32-bit) integers storing the number of buffers written for each active device. It has the format described in Table 3.6.5.

Table 3.6.5: Footer Format

Field	Type	Length
Buffer Counts	array[1 .. MaxStream] of long integers	32 bytes
filler	array[0..95] of char	96 bytes

The data buffer format is based on five word blocks consisting of a flag word followed by four data words as shown in Table 3.6.6.

Table 3.6.6: Data Buffer Format

WORD	5-WORD BLOCK			
1	1	2	3	4
2	1			
3	2			
4	3			
5	4			

The first word in a block contains flags in each nibble which define the content of the corresponding four words according to the table below. As Table 3.6.6 shows, there are four major word types. The data associated with Bus A and Bus B word types are simply the 16-bit value of the word which appeared on the bus (see Table 3.6.7).

Table 3.6.7: Data Associated with Bus A and Bus B Word Types

Bits 3:0 (hex)	Description	Modifier
0000	BUS A	CMD
0002		STS
0004		DATA
0001	BUS B	CMD
0003		STS
0005		DATA
0008	TIMER WORD	16-bit timer
0009		48-bit timer hi word
000A		48-bit timer mid word
000B		48-bit timer lo word
000C	FLAG WORD	stream number
000D		error flag word
000E		trigger event
000F		assoc word not used

When a new block is used, the flag word is initialized to FFFFh. As each new word is placed into the block, the associated flag value is shifted into the lower nibble of the flag word. If a block is only partially filled there will be flag entries of 000Fh remaining in the resulting flag word. The correspondence between flag entries and flag words in this case in shown in Table 3.6.8. This occurs when a timer block is inserted into the data stream (discussed below).

Table 3.6.8: Correspondence between Flag Entries and Flag Words

WORD	5-WORD BLOCK			
1	F	F	1	2
2	1			
3	2			
4	not used			
5	not used			

Each message is marked by a 48-bit timer word in the format shown in Table 3.6.9 and Table 3.6.10. This format is inserted whenever a command word is placed in the buffer or a bus switch occurs. To facilitate rapid storage of timer blocks, they are always placed on a 5-word block boundary. When a timer block is required, the remainder of the current block is skipped and the timer information (including a 16-bit gap time) is inserted in the next block.

Table 3.6.9: Timer Word Format for PC2 PASS

High	*****	**Days**		******	**Hours**	
	15	14	6	5	4	0
Middle	**Min**		**Sec**		**µSec**	
	15	10	9	4	3	0
Low	**µSeconds**					
	15					0

Table 3.6.10: Timer Word Format of PC3, PCI, and PCMCIA PASS

High	*****	******	**Days**		**Hours**	
	15	14	13	5	4	0
Middle	**Min**		**Sec**		**µSec**	
	15	10	9	4	3	0
Low	**µSeconds**					
	15					0

* 0 indicates board is synchronized with IRIG time source.
1 indicates board is free running.

** Bits not used

Flag words are used to indicate the occurrence of errors in the buffer. In most cases, the flag word is inserted in the buffer prior to the word that it modifies.

Cross Reference: The flag values are defined in the Error Table subsection of Appendix A.

3.7 Playback

The Playback feature reproduces the bus traffic recorded in a PASS-1000 archive file. This reproduction includes all BC commands and data, and intermessage and response gap times. By default, RT status responses and data are also included; however, they may be filtered from the archive file for exclusion from the playback operation.

During playback, all normal PASS-1000 RT emulation, RT view, and BC functions are disabled. PASS monitor functions are still accessible, however, allowing the playback results to be archived. Following the playback operation, the BC and RT setups are restored.

For playback, the PASS reproduces the information contained in binary archive files. Up to eight files may be played back at one time, but *each file must contain only one stream*. If necessary, convert a multiple stream archive file to separate single-stream files using the “Save as Binary File” option in the Data Logging **File** menu.

Cross Reference: For more details, see “Data Logging” on page 63.

When multiple files are played back simultaneously, they are started sequentially. Generally, depending on the number of files and the type of system, they all will be initialized within 10–20 microseconds.

3.7.1 Playback Control

Select **Playback** from the Main Menu to open the Playback Control window (shown in Figure 3.7.1). The window contains a menu bar, a list of devices, control buttons, and a status line.

Warning: Do not attempt to perform any other operation during a Playback session, including resizing a window, or switching to another Windows application. Any such operation will affect the reproduced data adversely.

A list of the available streams is displayed in the **Devices:** box. To assign a file for playback, highlight the desired device and click the Assign File button. When the File Open window appears, select the desired archive file and click **OK**. If the selected file is valid, the filename appears next to the device name in the **Devices:** box.

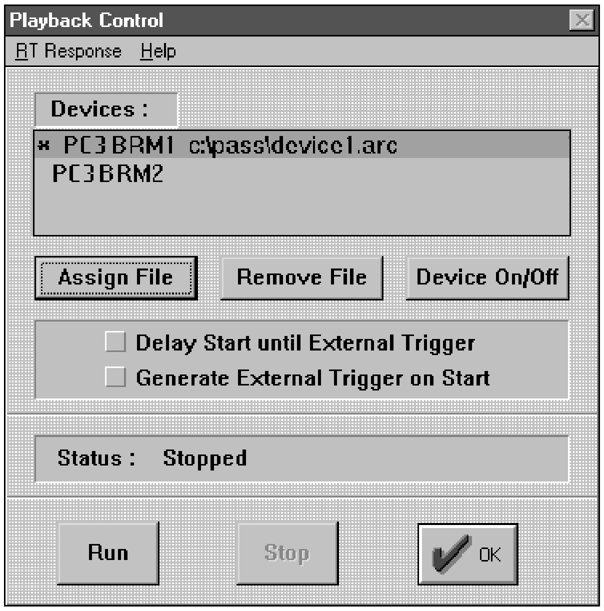


Figure 3.7.1: Playback Control Window

An asterisk appearing to the left of the device name indicates that the specified archive file is selected for playback. Toggle the asterisk on or off using the **Device On/Off** button. To deselect a file, highlight the desired device and click the Remove File button.

Select RT Responses from the menu bar to exclude specific RT responses from playback. Select a device to open the RT Responses window. Selected RTs appear in green; deselected RTs appear in grey. Click on an RT number to exclude it from playback. Click the **All** button to select all RTs; click the **None** button to deselect all RTs. When all RTs are deselected, the playback operation will consist of only a BC List.

External Triggers The Playback Control window contains the two triggering options described below. One of these options may be selected for each of the devices listed in the Playback Control window.

3.7.2 Delay Start until External Trigger

Highlight the desired PASS device and select the check box next to Delay Start until External Trigger to begin Playback after an external trigger is received by the device.

3.7.3 Generate External Trigger on Start

Highlight the desired PASS device and select the check box next to Generate External Trigger on Start to generate an external trigger signal from the device when Playback begins.

Run Playback When all the desired archive files have been assigned and selected, click the **Run** button to begin playback. If the Delay Start until External Trigger box is selected for a device, playback for that device will not begin until an external trigger signal is received. The Status line indicates the current state of the playback operation.

3.8 Utilities

Utilities are available to allow the user to view bus activity, view and modify PASS-1000 memory, display the status of the device, assign 10 character labels for each RT, set the protocol (1553A or 1553B) for each RT, define 1553A mode codes, adjust the voltage output level, and configure or display the ABI-PC2's internal time.

3.8.1 Bus Activity

The Bus Activity menu permits quick viewing of bus and error activity occurring on each RT address and its respective subaddress. Choose the Percent Display option to pop up the Bus Activity by RT window (shown in Figure 3.8.1). This window displays the activity for each RT as a percentage of the total bus activity using a bar chart and a value within a small display box.

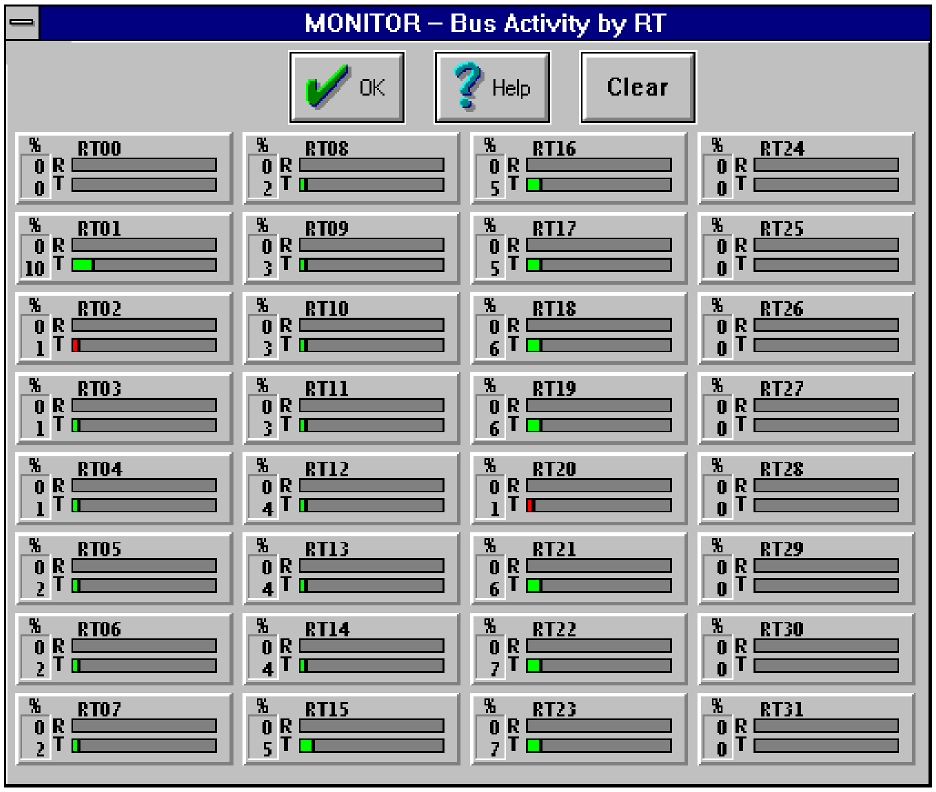


Figure 3.8.1: Bus Activity by RT Window

The window contains a display box for each RT address. Within each, there are two horizontal bar charts and two text boxes—one for receive and one for transmit subaddresses—that display the percentage of bus traffic associated with that RT. If an RT is experiencing errors on the bus, the bar chart will display a red bar; otherwise, it will be green. Due to rounding errors, the percentages for all RTs may or may not total 100%.

To display a summary of errors, place the mouse cursor over the RT's display box and press and hold down the *left* mouse button. The Error Summary panel will appear for that RT describing the protocol error(s) that occurred. To display a summary of bus activity by subaddress, place the mouse cursor over the RT's display box and click the *right* mouse button. The Bus Activity by SA window (shown in Figure 3.8.2) appears.

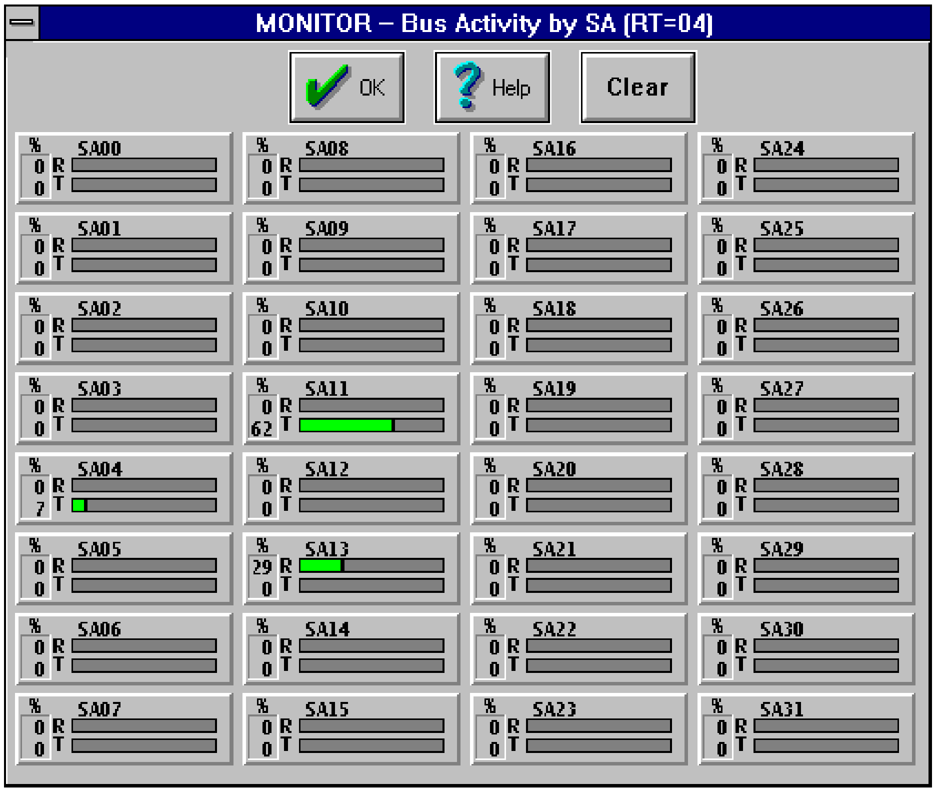


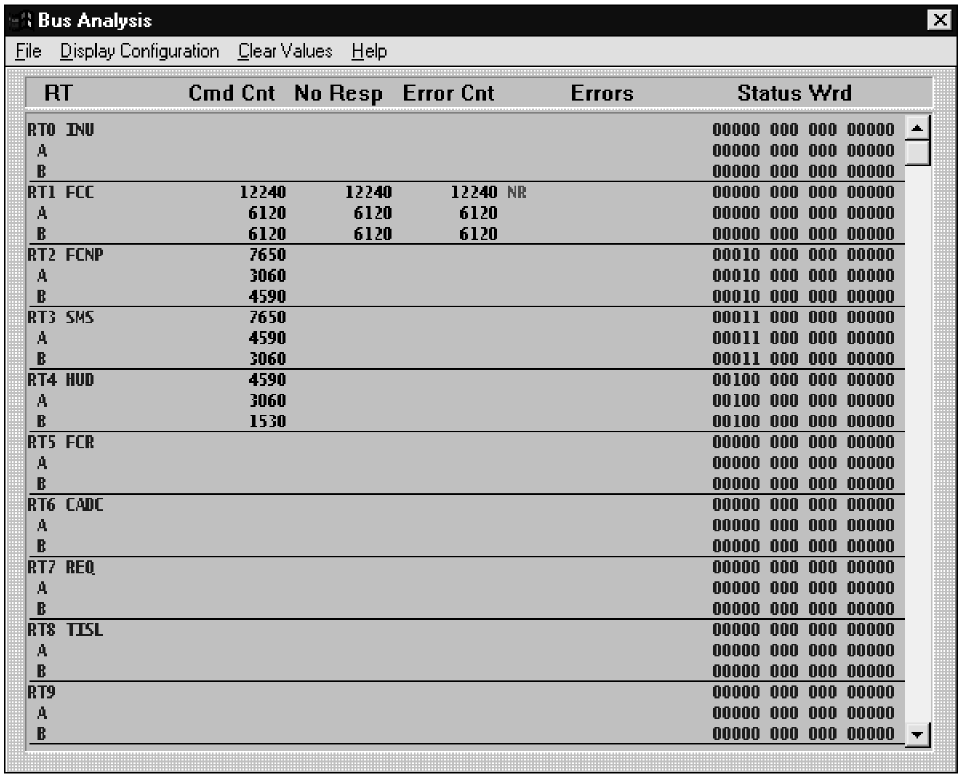
Figure 3.8.2: Bus Activity by SA window

This window is similar to the one displaying RT addresses except it contains a display box for each RT subaddress. Within each, there are two horizontal bar charts and two text boxes—one for receive and one for transmit subaddresses—that display the percentage of bus traffic associated with that subaddress number. If a subaddress has experienced errors on the bus since the last Clear, the bar chart will display a red bar; otherwise, it will be green. Due to rounding errors, the percentages for all subaddresses may or may not total to 100% of traffic directed to that RT. To display a summary of errors, place the mouse cursor over the subaddress's display box and press and hold down the left mouse button. The Error Summary panel will appear for that subaddress describing the protocol error(s) that occurred.

If there is no current activity, these windows will display the last activity recorded. The **Clear** button will clear past bus errors and activity.

3.8.2 Bus Analysis

Choose the Count Analysis option in the Bus Activity menu to pop up the Bus Analysis window (shown in Figure 3.8.3). This window displays running counts of commands and errors received by an RT. There are five columns which are being updated continuously while the monitor function is active. Data is listed by Bus A, Bus B, and combined for each RT. The data contained in each column is explained below.



```
RT              Cmd Cnt  No Resp  Error Cnt      Errors         Status Wrd
RT0 INU                                                  00000 000 000 00000
 A                                                       00000 000 000 00000
 B                                                       00000 000 000 00000
RT1 FCC           12240    12240    12240 NR             00000 000 000 00000
 A                 6120     6120     6120                00000 000 000 00000
 B                 6120     6120     6120                00000 000 000 00000
RT2 FCNP           7650                                  00010 000 000 00000
 A                 3060                                  00010 000 000 00000
 B                 4590                                  00010 000 000 00000
RT3 SMS            7650                                  00011 000 000 00000
 A                 4590                                  00011 000 000 00000
 B                 3060                                  00011 000 000 00000
RT4 HUD            4590                                  00100 000 000 00000
 A                 3060                                  00100 000 000 00000
 B                 1530                                  00100 000 000 00000
RT5 FCR                                                  00000 000 000 00000
 A                                                       00000 000 000 00000
 B                                                       00000 000 000 00000
RT6 CADC                                                 00000 000 000 00000
 A                                                       00000 000 000 00000
 B                                                       00000 000 000 00000
RT7 REQ                                                  00000 000 000 00000
 A                                                       00000 000 000 00000
 B                                                       00000 000 000 00000
RT8 TISL                                                 00000 000 000 00000
 A                                                       00000 000 000 00000
 B                                                       00000 000 000 00000
RT9                                                      00000 000 000 00000
 A                                                       00000 000 000 00000
 B                                                       00000 000 000 00000
```

Figure 3.8.3: Bus Analysis Window

Table 3.8.1: Bus Analysis Window Columns

Column	Description
Cmd Cnt	Displays a running count of all the commands received by this RT.
No Resp	Displays the number of No Response errors received by this RT.
Error Cnt	Displays the total number of errors received by this RT.
Errors	Lists an error code for each type of error received. See Appendix A for a description of each error code.
Status Word	Gives status response information for this RT, in a combined binary format. The five left-most bits display the RT from which the status response is received. The remaining bits display the actual status response word. The value displayed in this column is obtained by combining the new response with all previous responses in a logical OR operation.

Bus Analysis Menu Options

The menu bar in the Bus Analysis window provides options for customizing the display and for printing the contents of the screen.

File Menu Select the **Print Screen** option from this menu to print the data currently displayed in the Bus Analysis window on the default Windows printer. Select the **Exit** option from this menu to exit the Bus Analysis window.

Display Configuration Choose the **Display Configuration** option on the menu bar to pop up the Analysis Display Setup window (shown in Figure 3.8.4).

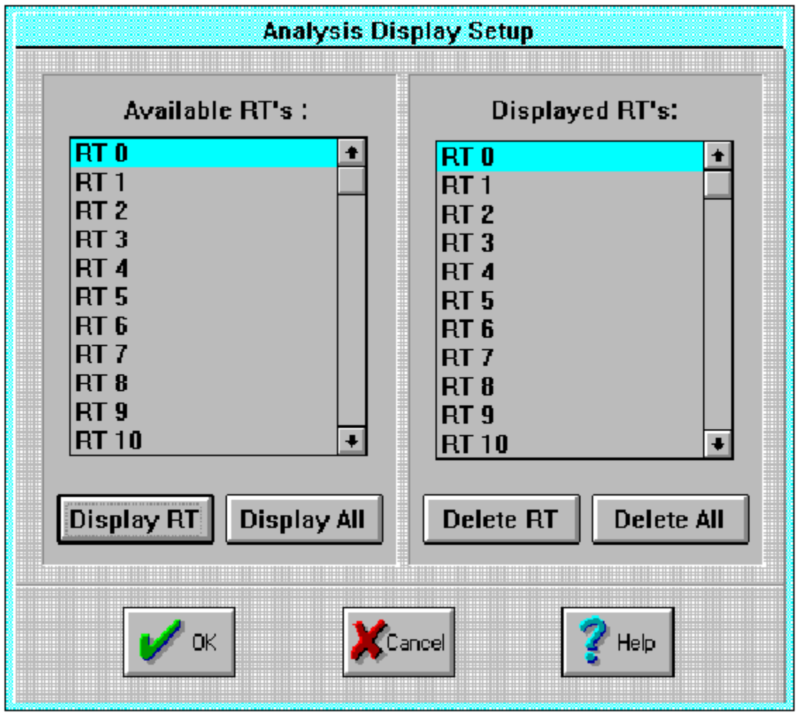


Figure 3.8.4: Analysis Display Setup Window

Use this window to customize the display. The RTs available for viewing are listed in the box on the left side of this window. The RTs currently being displayed are listed in the box on the right side of the window. The order of the RTs in the Displayed RT's list determines the order of the information displayed in the Count Analysis window. Use the mouse to highlight an RT in the left box, then click Display RT to add it to the list of Displayed RTs in the right box, below the highlighted entry. In the same way, highlight an RT in the right box, then click Delete RT to remove it from the Displayed RTs list. Click Display All to display all the available RTs in the Bus Analysis window. Click Delete All to remove all the RTs from the display window. Click OK to save the changes and exit from this window. Click Cancel to exit without saving.

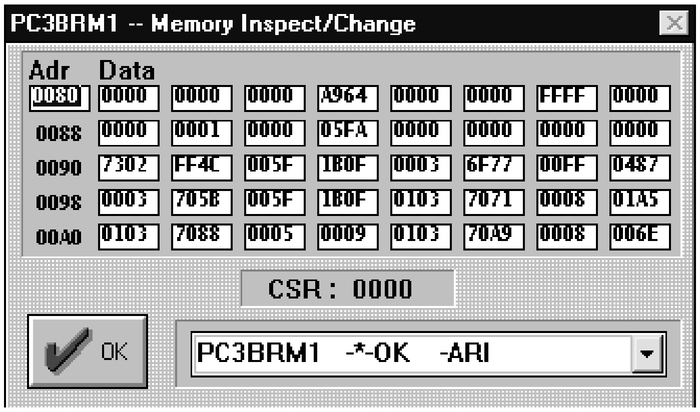
Clear Values Select the Clear Values option from the menu bar to set all the count values to zero.

Note: If the Clear Values option is selected while the bus is active, values for command, no response, and error may differ by one count due to differences in the times during the message at which they are calculated.

3.8.3 Tools Menu

Memory Inspect/Change

The Memory Inspect/Change window (shown in Figure 3.8.5) is intended primarily for test and diagnostic purposes. It permits the inspection or modification of any portion of the 64K words of memory onboard a device (ABI-PASS). From the Tools menu, choose Memory Inspect.



PC3BRM1 -- Memory Inspect/Change

Adr	Data							
0080	0000	0000	0000	A964	0000	0000	FFFF	0000
0088	0000	0001	0000	05FA	0000	0000	0000	0000
0090	7302	FF4C	005F	1B0F	0003	6F77	00FF	0487
0098	0003	705B	005F	1B0F	0103	7071	0008	01A5
00A0	0103	7088	0005	0009	0103	70A9	0008	006E

CSR : 0000

PC3BRM1 -*-OK -ARI

Figure 3.8.5: Memory Inspect/Change Window

This window displays 40 words of memory at a time. The left-most column displays the starting address of each row of data. The drop-down list box at the bottom of the window is used to change the memory inspect window from one device to another. The name in the Title Bar corresponds to the device selected in the Status Bar.

To change the starting address of the data being inspected, place the cursor over the word **Adr** text box, click or double-click the left mouse button, and enter the desired hexadecimal address.

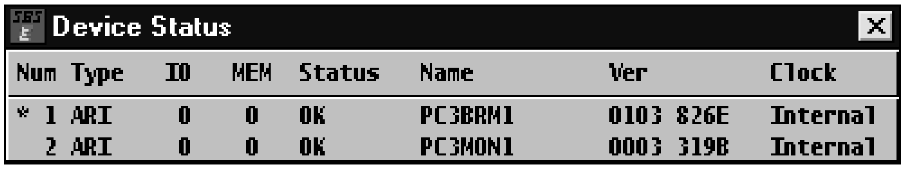
To change the contents of a data word, place the cursor over a data word text box, click or double-click the left mouse button, and enter the desired hexadecimal value. Pressing TAB will move the edit cursor to the next cell and highlight the value. At this point a new data word may be entered. Use the standard keyboard arrow keys to position the edit cursor and enter a new value. This process may be repeated as often as necessary.

Use the PAGE UP and UP ARROW keys to move forward in memory and the

PAGE DOWN and DOWN ARROW keys to move backwards.

Warning: This window is provided as a debugging tool. Changes to PASS-1000 memory locations during operation may cause unpredictable behavior.

Device Status From the Tools menu, choose **Device Status**. The Device Status window (shown in Figure 3.8.6) provides additional status information concerning the devices supported by the PASS-1000 and defined in the *PASS.CFG* file.



```
Num Type   IO   MEM  Status   Name        Ver        Clock
* 1 ARI     0     0  OK       PC3BRM1     0103 826E  Internal
  2 ARI     0     0  OK       PC3MON1     0003 319B  Internal
```

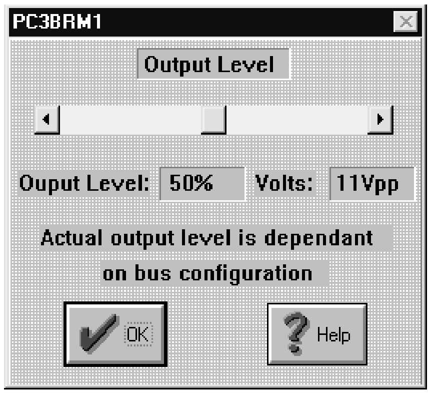
Figure 3.8.6: Device Status Window

Two lines of information are displayed for each ABI-PASS. The first line contains information about the BC and RT region of the board (denoted by an asterisk), and the second line specifies the monitor region. Each line contains the following information described in Table 3.8.2.

Table 3.8.2: Device Status Columns

Column	Description
Num	The *PASS.CFG* file contains two device definition entries for each ABI-PASS. This value reflects the order of these entries within the *PASS.CFG* file.
Type	Type of device installed in the Bus Analyzer. This value is defined in the *pass.cfg* file (see the “Installation” section of this manual).
IO	ABI-PASS base I/O address.
Mem	ABI-PASS memory base address.
INT	ABI-PASS interrupt level.
Status	Status of the device (OK or Not OK).
Name	Label name for the device. This value is defined in the *pass.cfg* file and corresponds to the device name displayed in Status Bar.
Ver	ABI firmware version number.
Clock	Source of the clock signal. If the PASS-1000 includes the External Clock opion, this field specifies Master or Slave. Otherwise, the clock source is Internal.

Output Level It is possible to adjust the amplitude of the signal transmitted by the PASS-1000. From the Tools menu, choose **Output Level** to display the Output Level window (shown in Figure 3.8.7). Move the scroll bar to select a percentage of the total output. The output may currently be adjusted from 7 to 22 volts (peak-to-peak, for ABI-PC3). The voltage value displayed is approximate. The actual output voltage is determined by bus configuration.



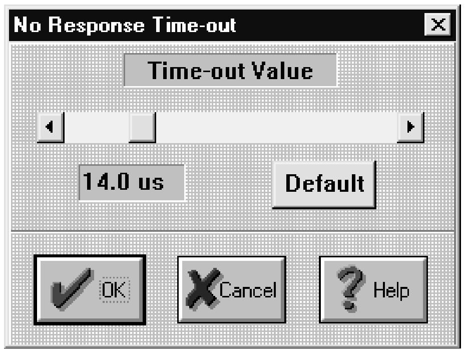
Ouput Level: 50% Volts: 11Vpp

Actual output level is dependant on bus configuration

Figure 3.8.7: Output Level Window

Set No Response Time-out

The response time-out value is the amount of time the ABI-PASS allows for an RT response before reporting a time-out error. Choose the **No Response Time Out Setting** option in the Tools menu to display the No Response Time-out window and to adjust this value from 7.0 to 40.0 microseconds (see Figure 3.8.8). Click on the scroll bar arrows to adjust the time-out value by 0.5 microseconds.

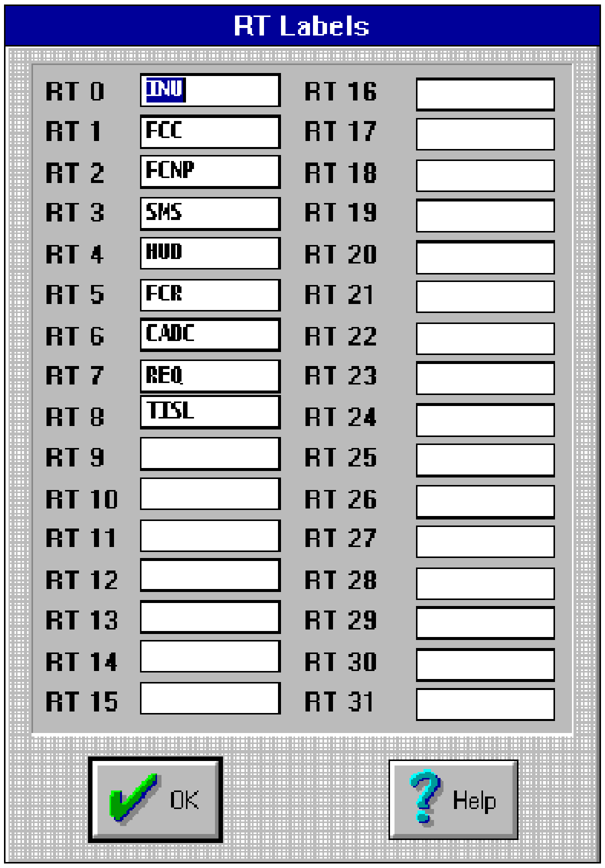


14.0 us

Figure 3.8.8: No Response Time-Out Window

RT Labels

The RT Label function in the Tool Menu allows the assignment of a name (of up to ten characters) to each RT (see Figure 3.8.9). This name will be displayed in each RT related dialog including the Bus Activity display, the RT define and view dialogs, and both Monitor displays. The assigned labels are saved with the **.BA3* files created by the File Save function and with the **.ARC* files created by the Data Logging. When previously created data log files are viewed using the data logging display, the RT Labels saved with the file will be displayed.



RT Labels

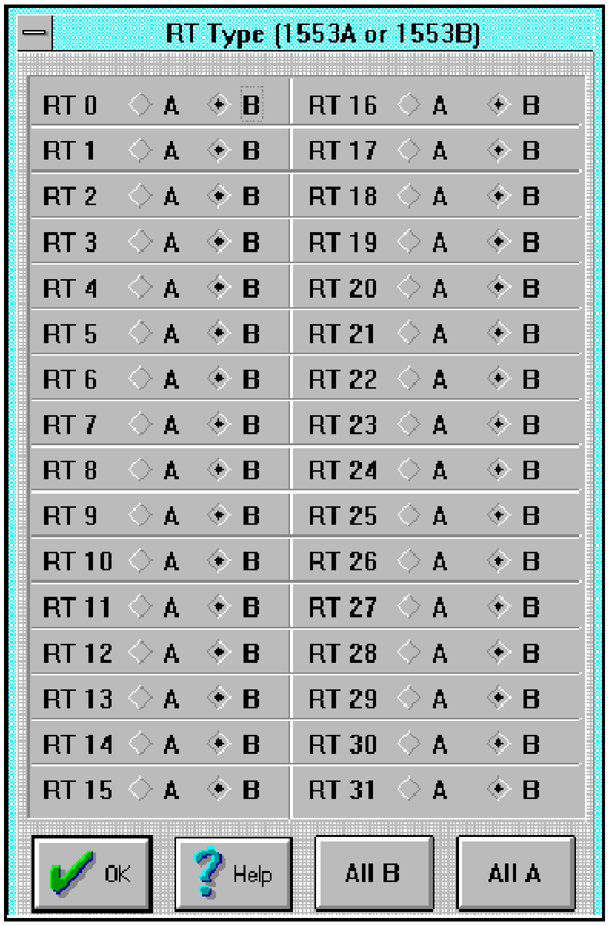
RT 0	INU	RT 16	
RT 1	FCC	RT 17	
RT 2	FCNP	RT 18	
RT 3	SMS	RT 19	
RT 4	HUD	RT 20	
RT 5	FCR	RT 21	
RT 6	CADC	RT 22	
RT 7	REQ	RT 23	
RT 8	TISL	RT 24	
RT 9		RT 25	
RT 10		RT 26	
RT 11		RT 27	
RT 12		RT 28	
RT 13		RT 29	
RT 14		RT 30	
RT 15		RT 31	

Figure 3.8.9: RT Labels Window

Labels are entered and changed by selecting the edit box corresponding to the correct RT and entering the desired label and ENTER or TAB.

Setting RT Protocol Types

Use the RT Protocol selection in the Tools pull down menu to set the protocol (1553A or 1553B) of individual RTs (see Figure 3.8.10). Click the **All B** or **All A** buttons to easily change the protocols of all the RTs. 1553A protocol RTs respond to user-defined mode codes and allow normal messages to subaddress 31. Save this RT protocol setting along with the setup files using the **Save All** option in the File pull down menu.



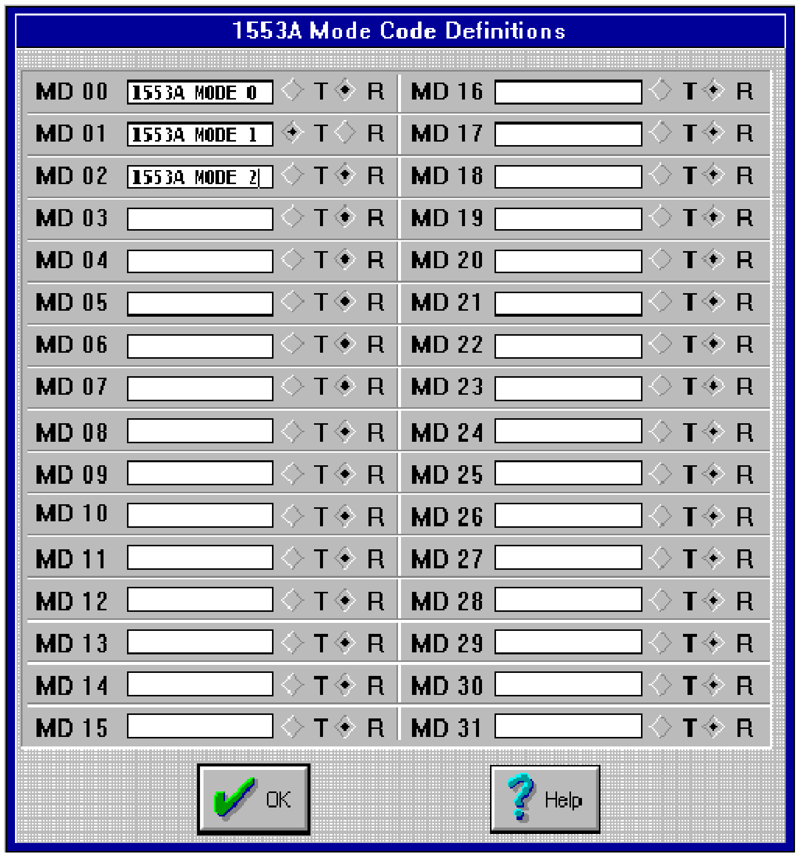
RT 0	A [✓ B]	RT 16	A [✓ B]
RT 1	A [✓ B]	RT 17	A [✓ B]
RT 2	A [✓ B]	RT 18	A [✓ B]
RT 3	A [✓ B]	RT 19	A [✓ B]
RT 4	A [✓ B]	RT 20	A [✓ B]
RT 5	A [✓ B]	RT 21	A [✓ B]
RT 6	A [✓ B]	RT 22	A [✓ B]
RT 7	A [✓ B]	RT 23	A [✓ B]
RT 8	A [✓ B]	RT 24	A [✓ B]
RT 9	A [✓ B]	RT 25	A [✓ B]
RT 10	A [✓ B]	RT 26	A [✓ B]
RT 11	A [✓ B]	RT 27	A [✓ B]
RT 12	A [✓ B]	RT 28	A [✓ B]
RT 13	A [✓ B]	RT 29	A [✓ B]
RT 14	A [✓ B]	RT 30	A [✓ B]
RT 15	A [✓ B]	RT 31	A [✓ B]

Figure 3.8.10: RT Protocol Window

Mode Code Definitions

Use the Mode Code Definition screen (shown in Figure 3.8.11) to assign up to 20 character descriptions for each 1553A mode code. Also use this dialog to determine whether a given mode code will be transmitted as a receive or transmit command (i.e., whether the 1553 transmit bit is set when the code is sent by the PASS-1000 bus controller list). The descriptions defined here are used by the PASS-1000 monitor functions to display mode codes sent to 1553A RTs. They are also used by the PASS-1000 bus controller functions to select and display mode codes that will be sent to 1553A RTs.

The mode code descriptions are saved in the **.BA3* files created using the **Save All** function and are restored using the **Load All** function in the File pull down menu. A copy of the descriptions are also saved in the **.ARC* files created by Data Logging. When these log files are viewed, the saved descriptions are used to display any mode codes sent to 1553A RTs during the logging period.



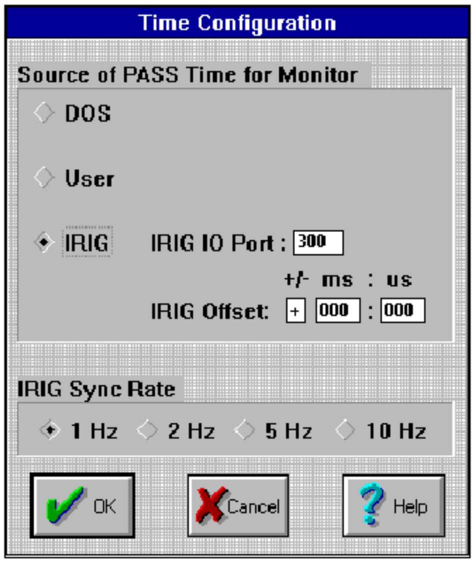
1553A Mode Code Definitions

MD 00	1553A MODE 0	T [✓ R]	MD 16		T [✓ R]
MD 01	1553A MODE 1	[✓ T] R	MD 17		T [✓ R]
MD 02	1553A MODE 2	T [✓ R]	MD 18		T [✓ R]
MD 03		T [✓ R]	MD 19		T [✓ R]
MD 04		T [✓ R]	MD 20		T [✓ R]
MD 05		T [✓ R]	MD 21		T [✓ R]
MD 06		T [✓ R]	MD 22		T [✓ R]
MD 07		T [✓ R]	MD 23		T [✓ R]
MD 08		T [✓ R]	MD 24		T [✓ R]
MD 09		T [✓ R]	MD 25		T [✓ R]
MD 10		T [✓ R]	MD 26		T [✓ R]
MD 11		T [✓ R]	MD 27		T [✓ R]
MD 12		T [✓ R]	MD 28		T [✓ R]
MD 13		T [✓ R]	MD 29		T [✓ R]
MD 14		T [✓ R]	MD 30		T [✓ R]
MD 15		T [✓ R]	MD 31		T [✓ R]

Figure 3.8.11: Mode Code Definitions Window

Time Source Configuration

The Time Configuration Dialog (shown in Figure 3.8.12) is used to select the time source that is used to generate message time stamps for both the Snapshot mode internal message buffer and the external Data Logging mode files.



Source of PASS Time for Monitor
- DOS
- User
- [✓ IRIG] IRIG IO Port : 300

+/- ms : us
IRIG Offset: + 000 : 000

IRIG Sync Rate
[✓ 1 Hz] 2 Hz 5 Hz 10 Hz

Figure 3.8.12: Time Configuration Window

The three available time options are described in Table 3.8.3.

Table 3.8.3: Time Options

Option	Description
DOS	This is the default option (unless IRIG is indicated in the *PASS.CFG* file). The ABI-PASS internal clock is initialized to the PC's DOS clock and runs without external updates unless the Time option is changed.
User	This option allows the user to select the initialization value for the ABI-PASS clock. If selected, edit boxes appear for entering the new time values. As with the DOS option, the ABI-PASS internal clock will run without updates until the time option is changed.
IRIG	This is the default option for devices that have the optional I parameter added to the device line in the *PASS.CFG* file. It is used when an external IRIG signal source is connected to the ABI card. If the *PASS.CFG* file contains the I parameter and an external IRIG signal is present, the DOS and User options are not valid. If the *PASS.CFG* file contains the I parameter and an external IRIG signal is not present prior to a monitoring operation, the DOS or User option may be selected.

If the external IRIG signal is lost while the PASS is monitoring data, the ABI-PASS internal clock "free-wheels" from the last IRIG time value. When this occurs, the time information in the monitor window is displayed in red.

The IRIG option is not available when the PASS-1000 is configured for the External Clock option.

Display Time Select **Display Time** to pop up a movable Window that displays the current value of the selected ABI-PASS time source. The value will reflect the source and setting provided in the **Time Source Configuration** dialog. This display is updated approximately every 0.5 seconds. If the IRIG option is chosen, an asterisk will be displayed to the left of the time value whenever the IRIG signal is not present.

About From the Tools menu, choose About to display the PASS-1000 software version number.

A: Error Reporting

Errors are recorded by the BC, Monitor, and Bus Activity functions. The BC reports errors by listing a message in the Status Box (of the BC Control Panel) and an error code in the BC text box of the Status Bar. The Monitor reports errors by displaying a character flag and a message beside the affected data word or message. The Count Analysis option in the Bus Activity menu displays the error types received by each RT. Errors are listed in Table A.0.1.

Table A.0.1: Error Table

Error Code	Type	Flag/ Message	Description
0001	Parity error	P	A parity error occurred in the received word.
0002	Manchester error	M	A manchester error occurred while receiving the word. This is simple a zero crossing not within the +/-150 nanosecond window.
0004	Bad sync	S	Bit transitions occurred with no valid sync.
0008	Data lost	1.00	A new word was received before a pending word was read.
0010	Too many bits in word	B	A valid zero crossing occurred at the mid-bit time following the parity bit.
0020	Too few bits in word	b	The word terminated before receiving the required number of bits. Termination occurs when a zero crossing does not occur during mid-bit time for any bit.
0040	Unexpected gap	9.00	An unexpected gap occurred between words on the bus.
0080	Sync type error	C	A data word was expected but a contiguous word was received with a command sync.
0100	Too many words in message	2.00	More than the indicated number of words have been received with no intervening gap.
0200	Too few words in message	3.00	A gap occurred in a message when data was expected.
0400	No gap	4.00	A gap was expected but did not occur between words on the bus.
0800	Illegal 1553B mode code	5.00	An illegal mode command occurred.

1000	Word count error (RT-RT word count discrepancy)	6.00	The word count field didn't match between two RT-RT commands. This error should follow the second command word in the buffer.
4000	Status response time-out	7.00	A status word is expected but not received before a gap timeout.
8000	Spurious data	8.00	Unexpected data appeared on the bus. This can be caused by an invalid or corrupted command word. It is often associated with Error Code 0100.

The Error Code is only reported in the BC text box of the Status Bar. If there are multiple errors, each Error Code is added together to form one code. For example, an Error Code equal to 4001 indicates a parity error and status response timeout.

Errors are classified as word or message errors. Word errors affect a single data word in the data stream. One example is a parity error. Message errors affect the entire message. Examples of this class of error include a sync error or the wrong number of data words transmitted.

The Flag/Message column indicates whether the error is a word or message error. A letter indicates a word error (See Table 0.1.1) and a numeral indicates a message error (See Table 0.2.1).

A.1 Word Errors

Word errors are designated by a single letter which follows a data, gap, status, or command word displayed in the monitor buffer. These designations are listed below and referenced in the Error Table.

Table 0.1.1: Word Error Designations

Error Type	Letter
Parity Error	P
Manchester	M
Too many bits in word	B
Too few bits in word	b
No valid sync character	S
Data word with command sync	C

A.2 Message Errors

Message errors appear in red on a line of text in the monitor buffer to describe the messages error.

Table 0.2.1: Message Error Designations

No.	Error Message
1	Data Lost
2	Too many words in message!
3	Too few words in message!
4	Expected Gap not Found
5	Illegal Mode Command
6	Word Count Discrepancy
7	Status response timeout!
8	Spurious Data!
9	Unexpected Gap

A.3 Error Codes in Count Analysis

In the Count Analysis option of the Bus Analysis menu, error types are displayed in a code format. These codes are listed in Table 0.3.1.

Table 0.3.1: Count Analysis Error Codes

Code	Error Type
P	Parity Error
M	Manchester Error
>B	Too Many Bits In Word
<B	Too Few Bits In Word
BS	Bad Sync
CS	Command Sync Error
DL	Data Lost
SD	Spurious Data
NR	Status Response Timeout
>W	Too Many Words In Message
<W	Too Few Words In Message
NG	Expected Gap Not Found
IM	Illegal Mode Command
WC	Word Count Discrepancy

B: ASCII File Formats

The Save as ASCII option in the PASS-1000 File menu will save the BC List, RT Setup, or Misc. Setup to an ASCII file. The Load ASCII File option in the File menu will read a BC, RT, or Misc. ASCII save file and load the PASS-1000 with the data stored in the file. These ASCII files may be edited using any spreadsheet software and most word processing software.

Note: Word processing software used to edit a PASS ASCII file must *not* expand or delete tabs.

This appendix details the formats of the BC, RT, and Misc. ASCII files. Examples are given for each of these file types, as well as detailed descriptions of each key word. Rather than creating an entire configuration in an ASCII format, SBS recommends first building a basic BC List, RT Setup, or Misc. Setup using the PASS-1000 software, saving it as an ASCII file, and then editing the file to change data or add messages.

The data fields in the PASS ASCII files are separated by tabs (tab delineated). Use caution when editing the files because the PASS software will be unable to correctly read the data if these tabs are removed or altered.

B.1 BC List

The information in a BC List ASCII file is structured in a table format, having up to eleven columns. Each line contains a key word in the first column which designates it as a BC message, data, error, or minor frame timer. BC message types are each designated by a specific key word, such as BC-RT, RT-RT, or Mode. Data, error, and minor frame entries are designated by the key words Data, Error, and Frame, respectively. Each message definition is followed by a blank line (carriage return -line feed sequence).

For each of the following format descriptions, refer to Example of BC List File.

B.1.1 Defining a Minor Frame

The BC List may be broken up into minor frames through the use of Minor Start entries. These entries are defined in the ASCII file by the key word Frame (first column), followed by the number of microseconds (second column). By default, each message list starts with a Frame Start message.

B.1.2 BC Message Format

Message formats are as follows:

Key word	Bus	RT#	R/T	SA#	WC	Time	Hex Cmd.

Key word	This field designates the type of message transfer. Valid types are:	
	BC-RT	Bus Controller to Remote Terminal
	RT-BC	Remote Terminal to Bus Controller
	RT-RT1	Remote Terminal to Remote Terminal (Receive)
	RT-RT2	Remote Terminal to Remote Terminal (Transmit)
	Mode	Mode Commands
Bus	This field specifies the bus (A or B) that will transmit the command.	
RT#	This field identifies the Remote Terminal number.	
R/T	This field specifies the direction of the message: **R** for Receive and **T** for Transmit.	
SA#	This field identifies the subaddress number.	
WC	This field specifies the word count for the message. For **Mode** message types, this field specifies the type of mode command.	
Time	This field specifies the intermessage gap time, in microseconds. This field is not used for **RT-RT2** or **Mode** message types.	
Hex Cmd.	This field, enclosed by parentheses, contains a hexadecimal representation of the command word. This is an optional field for reference only; it is not used by the PASS software.	

B.1.3 Data Entries

Data entries directly follow **BC-RT** messages and certain **Mode** commands. Each data entry contains a maximum of ten binary data words, separated by tabs. Data is in the following format:

Data	1	2	3	4	5	6	7	8	9	10

B.1.4 Wait for External Trigger

The External Trigger function suspends BC list transmission until a signal is received on the TRIGGER line of the PASS cable adapter. The PASS performs this function when it encounters a "Wait for External Signal" command in the BC list. This command is designated as **`Ext_Wait`** in the ASCII file.

B.1.5 Comment Lines

Non-executable comment statements may be added to the ASCII file. Comments must be preceded by a semicolon (;).

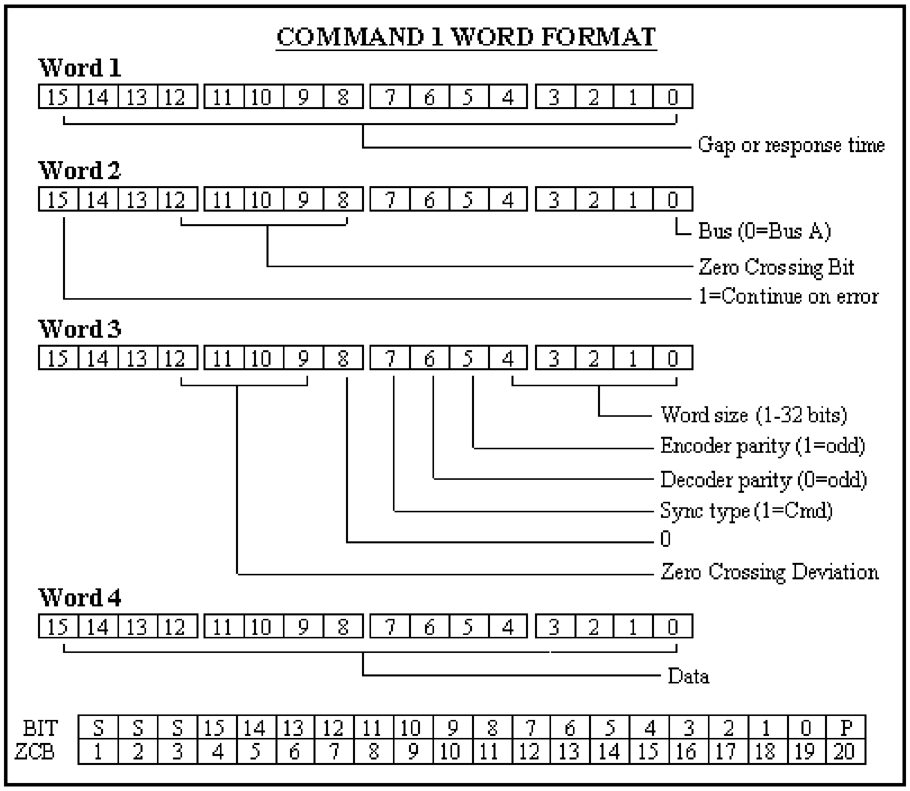
B.1.6 Error Entries

Error entries directly follow the message on which they occurred. They have the following format:

Error	Word	EW1	EW2

	This key word identifies this line as an error entry.	
Error	Word	This field specifies the word on which the error occurred:
	0	Error occurred on the command word
1-33	Error occurred on the indicated data word	
EW1	This field contains the first error word, preceded by **`0X`** (hexadecimal).	
EW2	This field contains the second error word, preceded by **`0X`** (hexadecimal).	

The bit definitions for the Error fields are determined by whether the error occurred on the command word or one of the data words. If the error occurred on the command word, the Command 1 Word Format applies, where ***EW1*** corresponds to Word 2 and ***EW2*** corresponds to Word 3. Words 1 and 4 do not apply. If the error occurred on a data word, the Data Word - Long Def Format applies, where ***EW1*** corresponds to Word 1 and ***EW2*** corresponds to Word 2. Word 3 does not apply. See the following pages for these bit definitions.



BIT	S	S	S	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	P
ZCB	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

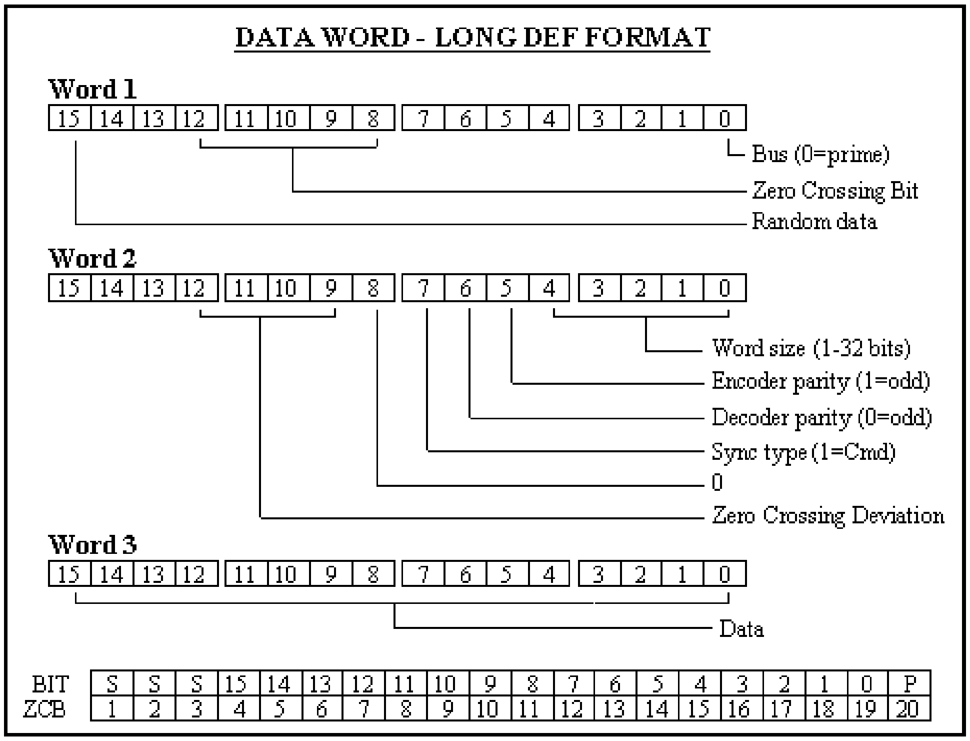
Word 1 specifies the intermessage gap or status response time for the command. Words 2 and 3 specify attributes of the word to be transmitted. Word 4 is simply the 16-bit command word to be transmitted.

In word 2, bit 0 specifies the bus on which the message was transmitted. Bit 15 indicates whether the remainder of the message was transmitted following the error. The value used in the Zero Crossing bit field is selected from the diagram at the bottom of the figure. For example, a binary 2 in this field affects the zero crossing in the Sync while a binary 6 affects data bit 13.

In word 3, the word size field (bits 0-4) is typically filled with a 13h (wordsize-1) for a 20-bit word size. Correct MIL-STD-1553 parity (bits 5 and 6) is odd. The following table lists the values used in the Zero Crossing Deviation field (bits 9-12).

Zero Crossing Deviation

Value	Effect (ns)
0x8	not used
0x9	not used
0xA	-375
0xB	-312.5
0xC	-250
0xD	-187.5
0xE	-125
0xF	-62.5
0x0	No deviation
0x1	62.5
0x2	125
0x3	187.5
0x4	250
0x5	312.5
0x6	not used
0x7	not used



This format is used when errors are inserted in a data word. In general, the bits function the same as those in the command word format with the exception of the Random data bit (word 1, bit 15). The Random data bit is not present in the command word specification because there is no benefit to transmitting random commands. This bit causes the PASS to ignore the data word in the buffer for this word and insert a pseudo random data word.

B.2 Example of BC List File

```
;    BC List

;              Usec
Frame     8000

;         Bus   RT     T/R     SA    Count Gap    Hex Cmd.
BC-RT     A     1      R       1     32    10.0   (0X0820)
Data      1     2      3       4     5     6      7        8     9     10
Data      11    12     13      14    15    16     17       18    19    20
Data      21    22     23      24    25    26     27       28    29    30
Data      31    32
;               Word   EW1     EW2
Error     4     0X0000 0X0013

;               Bus    RT      T/R   Mode  Count  Gap      Hex Cmd.
Mode      A     1      T       0     2     10.0   (0X0C02)

;               Bus    RT      T/R   SA    Count  Gap      Hex Cmd.
RT-BC     A     2      T       2     32    10.0   (0X1440)
```

```
;                Bus      RT       T/R   Mode  Count   Gap       Hex Cmd.
Mode      A      2        T        0     18    10.0    (0X1412)

;                Bus      RT       T/R   SA    Count   Gap       Hex Cmd.
RT-RT1    A      3        R        3     32    10.0    (0X1860)
RT-RT2    A      4        T        5     32            (0X1860)

;                Bus      RT       T/R   Mode  Count   Gap       Hex Cmd.
Mode      A      3        R        0     17    10.0    (0X1811)
Data      0

;                Bus      RT       T/R   SA    Count   Gap       Hex Cmd.
BC-RT     B      1        R        1     32    10.0    (0X0820)
Data      1      2        3        4     5     6       7         8     9     10
Data      11     12       13       14    15    16      17        18    19    20
Data      21     22       23       24    25    26      27        28    29    30
Data      31     32

;                Bus      RT       T/R   Mode  Count   Gap       Hex Cmd.
Mode      B      1        T        0     2     10.0    (0X0C02)

;                Bus      RT       T/R   SA    Count   Gap       Hex Cmd.
RT-BC     B      2        T        2     32    10.0    (0X1440)

;                Bus      RT       T/R   Mode  Count   Gap       Hex Cmd.
Mode      B      2        T        0     18    10.0    (0X1412)

;                Bus      RT       T/R   SA    Count   Gap       Hex Cmd.
RT-RT1    B      3        R        3     32    10.0    (0X1860)
RT-RT2    B      4        T        5     32            (0X1860)

;                Bus      RT       T/R   Mode  Count   Gap       Hex Cmd.
Mode      B      3        R        0     17    10.0    (0X1811)
Data      0

;                Bus      RT       T/R   SA    Count   Gap       Hex Cmd.
BC-RT     A      1        R        1     32    10.0    (0X0820)
Data      1      2        3        4     5     6       7         8     9     10
Data      11     12       13       14    15    16      17        18    19    20
Data      21     22       23       24    25    26      27        28    29    30
Data      31     32

;                Bus      RT       T/R   SA    Count   Gap       Hex Cmd.
BC-RT     B      1        R        1     32    10.0    (0X0820)
Data      1      2        3        4     5     6       7         8     9     10
Data      11     12       13       14    15    16      17        18    19    20
Data      21     22       23       24    25    26      27        28    29    30
Data      31     32

;                Bus      RT       T/R   Mode  Count   Gap       Hex Cmd.
Mode      A      1        T        0     2     10.0    (0X0C02)

;                Bus      RT       T/R   Mode  Count   Gap       Hex Cmd.
Mode      B      1        T        0     2     10.0    (0X0C02)

;                Bus      RT       T/R   SA    Count   Gap       Hex Cmd.
RT-RT1    A      3        R        3     32    10.0    (0X1860)
RT-RT2    A      4        T        5     32            (0X1860)

Ext_Wait

;                Bus      RT       T/R   SA    Count   Gap       Hex Cmd.
RT-BC     B      2        T        2     32    10.0    (0X1440)
```

B.3 RT Setup

The definition of each RT in the ASCII file consists of three sections: (1) heading, (2) receive subaddress definitions, and (3) transmit subaddress definitions. Each of these sections is structured in a table format, with tabs separating the columns. These sections are defined in the following paragraphs.

For each of the following format descriptions, refer to the "Example of RT Setup File" found in this appendix.

B.3.1 Heading

The heading portion of the RT definition identifies the RT number and defines any mode commands. The first line has the following format:

RTDEF	RT#	E/M	A/B

RTDEF	This key word identifies this line as the beginning of the RT definition.
RT#	This field identifies the RT number.
E/M	This field specifies whether the RT is being emulated (E) or map monitored (M).
A/B	This field specifies whether the RT is defined as 1553A or 1553B.

If mode commands are defined, they follow the RTDEF entry and have the following format:

Name	Time	Error1	Error2	Value

Name	This key word identifies the mode command. Valid entries are:	
	BIT	Transmit Bit Word
	LAST_COMMAND	Last Command Data
	VECTOR	Transmit Vector Word
	LAST_STATUS	Last Status Word
	GENERIC	
Time	This field specifies the status word response time, in microseconds.	
Error1	This field specifies the first error word, preceded by **0X** (hexadecimal).	

Error2	This field specifies the second error word, preceded by **0X** (hexadecimal).
Value	This field specifies the error data word.

The bit definitions for these fields follow the Command 1 Word Format, where ***Time*** corresponds to Word 1, ***Error1*** corresponds to Word 2, ***Error2*** corresponds to Word 3, and ***Value*** corresponds to Word 4. See the BC List ASCII File description for details.

B.3.2 Receive Subaddress Definitions

The key word SADEF designates the beginning of the subaddress definition section. Following this word, each receive subaddress is defined on a separate line. Only those subaddresses which are defined for this RT have an entry in the ASCII file. The receive subaddress definitions have the following format:

SAR	SA#	Time	Status	Bit field (optional)

SAR	This key word identifies this line as a receive subaddress definition.
SA#	This field specifies the subaddress number.
Time	This field specifies the status response time, in microseconds.
Status	This field contains the status word, preceded by "0X" (hexadecimal).
Bit field	This field, enclosed by parentheses, contains a bit representation of the status word. This is an optional field for reference only; it is not used by the PASS software.

B.3.3 Transmit Subaddress Definitions

Transmit subaddresses are defined immediately following the receive subaddress definitions. They have the following format:

SAT	SA#	WC Var	Time	Status	Bit field (optional)

SAT	This key word identifies this line as a transmit subaddress definition.

SA#	This field specifies the subaddress number.	
WC Var	This field determines the variation from the word count. Valid values are as follows:	
	0	Transmit exact number of words
	1	Transmit one additional word
	2	Transmit one less word
Time	This field specifies the status response time, in microseconds.	
Status	This field contains the status word, preceded by "0X" (hexadecimal).	
Bit field	This field, enclosed by parentheses, contains a bit representation of the status word. This is an optional field for reference only; it is not used by the PASS software.	

Thirty-three data words follow each transmit subaddress definition. Each line of data begins in the second column (preceded only by a tab), and contains a maximum of ten words per line. Any data word may be replaced by random data by entering Ran in its place.

Subaddress error entries have formats identical to those described for the BC List ASCII files, with one exception: the word field contains a zero if an error occurred on the status word.

B.3.4 Comment Lines

Non-executable comment statements may be added to the ASCII file. Comments must be preceded by a semicolon (;).

B.3.5 Ending Key Words

The key word RTDEF_END designates the end of the subaddress definitions for this RT. The key word END designates the end of the RT definition block.

B.4 Example of RT Setup File

```
;      RT Setup File

;      RT Definition
;        RT      Map/Emul1553B/A
RTDEF   1       E       B
;Mode   Gap      Word1  Word2   Value
BIT     6.5     0X0000  0X0033  0XDEAD
;Mode   Gap     Word1   Word2   Value
VECTOR  6.5     0X0000  0X0033  0XDEAF
SADEF
```

```
;     Receive Sub-addresses
;       SA      Gap     Status  Binary Status
SAR     1        6.5    0X0800  (00001 000 000 00000)
SAR     2        6.5    0X0800  (00001 000 000 00000)
;     Transmit Sub-addresses
;       SA      CntErr  Gap     Status  Binary Status
SAT     1       0        6.5    0X0800  (00001 000 000 00000)
3104    3105    3106    3107    3108    3109    3110    3111    3112    3113
3114    3115    3116    3117    3118    3119    3120    3121    3122    3123
3124    3125    3126    3127    3128    3129    3130    3131    3132    3133
3134    3135    3136
;               SA      CntErr  Gap     Status  Binary Status
SAT     2       0        6.5    0X0800  (00001 000 000 00000)
3136    3137    3138    3139    3140    3141    3142    3143    3144    3145
3146    3147    3148    3149    3150    3151    3152    3153    3154    3155
3156    3157    3158    3159    3160    3161    3162    3163    3164    3165
3166    3167    3168
RTDEF_END

;     RT Definition
;               RT    Map/Emul15553B/A
RTDEF   2       E       B
SADEF
;     Receive Sub-addresses
;       SA      Gap     Status  Binary Status
SAR     1        6.5    0X1000  (00010 000 000 00000)
SAR     2        6.5    0X1000  (00010 000 000 00000)
RTDEF_END

;     RT Definition
;               RT     Map/Emul15553B/A
RTDEF   30      E       B
;Mode   Gap     Word1   Word2   Value
GENERIC 6.5     0X0000  0X00B3  0XF000
SADEF
;     Transmit Sub-addresses
;       SA      CntErr  Gap     Status  Binary Status
SAT     1       0        6.5    0XF000  (00010 000 000 00000)
62496   62497   62498   62499   62500   62501   62502   62503   62504   62505
62506   62507   62508   62509   62510   62511   62512   62513   62514   62515
62516   62517   62518   62519   62520   62521   62522   62523   62524   62525
62526   62527   62528
;               SA      CntErr  Gap     Status  Binary Status
SAT     2       0        6.5    0XF000  (00010 000 000 00000)
62528   62529   62530   62531   62532   2533    62534   62535   62536   62537
62538   62539   62540   62541   62542   2543    62544   62545   62546   62547
62548   62549   62550   62551   62552   62553   62554   62555   62556   62557
62558   62559   62560
RTDEF_END
END
```

B.5Misc. Setup

The Misc. Setup ASCII file consists of six sections: (1) Output Level, (2) Timeout, (3) 1553A Mode Code Definitions, (4) IRIG/Time Source, (5) Filters, and (6) Triggers. Each of these sections is structured in a table format, with tabs separating the columns. These sections are defined in the following paragraphs.

For each of the following format descriptions, refer to the "Example of Misc.

Setup File" found in this appendix.

B.5.1 Output Level

This section specifies the output voltage for the PASS hardware. It consists of a heading on one line with a numeric value on the following line.

OUTPUT_LEVEL	This heading identifies this line as the beginning of the Output Level section.
NNN	This decimal value is located directly under the heading. Divide this value by 22 to obtain the output value in volts. This is only an approximation, however, and the actual output voltage may vary due to the amount of bus loading.

B.5.2 Timeout

This section specifies the no response timeout value. It consists of a heading on one line with a numeric value on the following line.

TIMEOUT	This heading identifies this line as the beginning of the Timeout section.
NN.N	This decimal value is located directly under the heading. It specifies the no response timeout in microseconds.

B.5.3 1553A Mode Code Definitions

This section details the user specified descriptions for 1553A mode codes. It consists of a heading followed by one line for each 1553A mode code description. Only those 1553A mode codes which have been assigned descriptions appear in the ASCII file.

MODE_A	This heading identifies this line as the beginning of the 1553A Mode Code section. mode T/R name
mode	This field identifies the mode code number.
T/R	This field specifies whether this mode code will be transmitted as a receive (R) or transmit (T) command.
name	This field contains the user defined description for this mode code.

B.5.4 Filters

This section details the receive and transmit subaddress filters. It consists of a heading and an Error Only specification, followed by one line for each RT. The subaddresses for each RT are divided into four columns, described below.

FILTERS	This heading identifies this line as the beginning of the Filters section.
ERROR_ONLY	If this line contains **`ERROR_ONLY_ON`**, only errors will be sent to the selected RT-SAs. If this line contains **`ERROR_ONLY_OFF`**, all messages will be sent to the selected RT-SAs.

RT	**R SA 15-0**	**R SA 31-16**	**T SA 15-0**	**T SA 31-16**

RT	This field contains the RT number.
R SA 15-0	This field contains 16 bits, corresponding to receive subaddresses 0-15. The left-most bit corresponds to subaddress 15. If a bit is set equal to "1", the filter for the corresponding subaddress is off (all messages directed to that subaddress will be stored in the monitor buffer).
R SA 31-16	This field contains 16 bits, corresponding to receive subaddresses 31-16. The left-most bit corresponds to subaddress 31. If a bit is set equal to "1", the filter for the corresponding subaddress is off (all messages directed to that subaddress will be stored in the monitor buffer).
T SA 15-0	This field contains 16 bits, corresponding to transmit subaddresses 0-15. The left-most bit corresponds to subaddress 15. If a bit is set equal to "1", the filter for the corresponding subaddress is off (all messages directed to that subaddress will be stored in the monitor buffer).
T SA 31-16	This field contains 16 bits, corresponding to transmit subaddresses 31-16. The left-most bit corresponds to subaddress 31. If a bit is set equal to "1", the filter for the corresponding subaddress is off (all messages directed to that subaddress will be stored in the monitor buffer).

B.5.5 Triggers

This section details the sixteen possible trigger blocks.

TRIGGERS	This heading identifies this line as the beginning of the Triggers definition.

0XNNNN	location	type

Note: The line above immediately follows the heading.

0XNNNN	This field identifies the block on which to jump after detecting the End of Message. The block number is preceded by **`0X`** (hexadecimal).

location	This field specifies where to locate the trigger message in the monitor buffer. Valid options are: S Start M Middle E End
type	This field indicates whether this is a SIMPLE or COMPLEX trigger type.
Block	Exp. Mask Value Count Goto1 Goto2 Goto3 Flag
Block	This field contains the trigger block number, 0-15.
Exp.	This field contains a four digit value (in hexadecimal) which specifies the bus (A or B) and the expression. The left-most digit indicates the bus: 0 Either Bus 2 Bus A Only 6 Bus B Only The right-most digit specifies the expression.Valid types are as follows: 0 Trigger 1 Command/= 2 Command/<> 3 Status/= 4 Status/<> 5 Data/= 6 Data/<> 7 Error/= 8 Error/<>
Mask	This field contains the mask value, in hexadecimal.
Value	This field contains the value to be compared with the masked word, in hexadecimal.
Count	This field contains the initial count value, in hexadecimal.
Goto1	This field specifies the block to be executed when the the first "if" statement is true.
Goto2	This field specifies the block to be executed when the second "if" statement is true.
Goto3	This field specifies the block to be executed when the third "if" statement is true.
Flag	This field identifies the equivalent simple trigger type.Valid types are as follows: 0 Trigger on any msg 1 Trigger on msg=MSGID

2	Trigger on msg=MSGID when MASKed data word NN = VALUE
3	Trigger on msg=MSGID when MASKed data word NN <> VALUE
4	Trigger on msg=MSGID when MASKed status word = VALUE
5	Trigger on msg=MSGID when MASKed status word <> VALUE
6	Trigger on any error
7	Trigger on msg=MSGID with any error
8	Trigger on msg=MSGID with exact error combo= ERROR
9	Trigger on msg=MSGID with "and" of selected errors = ERROR
10	Trigger on msg=MSGID with "or" of selected errors = ERROR
11	Trigger on msg=MSGID with error code <> ERROR

B.5.6 Comment Lines

Non-executable comment statements may be added to the ASCII file. Comments must be preceded by a semicolon (;).

B.6 Example Of Misc. Setup File

```
; Misc. Setup File

OUTPUT_LEVEL
128

TIMEOUT
14.0

MODE_A
;Mode     T/R       Name
0         R         mode_0
1         T         mode_1
2         R         a_weird_mode
7         T
21        T
22        T
31        T         last_mode

IRIG
;source   Base      Rate          Correction
D         0X0300    10            500000

FILTERS
ERROR_ONLY_ON
;RT   R SA 15-0          R SA 31-16                          T SA 15-0   T SA
31-16
0     0000000000000000   0000000000000000   0000000000000000   0000000000000000
1     0000000000000001   0000000000000001   0000000000000001   0000000000000001
2     0000000000000000   0000000000000000   0000000000000000   0000000000000000
3     0000000000000000   0000000000000000   0000000000000000   0000000000000000
4     0000000000000000   0000000000000000   0000000000000000   0000000000000000
5     0000000000000000   0000000000000000   0000000000000000   0000000000000000
6     0000000000000000   0000000000000000   0000000000000000   0000000000000000
7     0000000000000000   0000000000000000   0000000000000000   0000000000000000
8     0000000000000000   0000000000000000   0000000000000000   0000000000000000
9     0000000000000000   0000000000000000   0000000000000000   0000000000000000
10    0000000000000000   0000000000000000   0000000000000000   0000000000000000
11    0000000000000000   0000000000000000   0000000000000000   0000000000000000
12    0000000000000000   0000000000000000   0000000000000000   0000000000000000
13    0000000000000000   0000000000000000   0000000000000000   0000000000000000
14    0000000000000000   0000000000000000   0000000000000000   0000000000000000
15    0000000000000000   0000000000000000   0000000000000000   0000000000000000
16    0000000000000000   0000000000000000   0000000000000000   0000000000000000
17    0000000000000000   0000000000000000   0000000000000000   0000000000000000
18    0000000000000000   0000000000000000   0000000000000000   0000000000000000
19    0000000000000000   0000000000000000   0000000000000000   0000000000000000
20    0000000000000000   0000000000000000   0000000000000000   0000000000000000
21    0000000000000000   0000000000000000   0000000000000000   0000000000000000
22    0000000000000000   0000000000000000   0000000000000000   0000000000000000
23    0000000000000000   0000000000000000   0000000000000000   0000000000000000
24    0000000000000000   0000000000000000   0000000000000000   0000000000000000
25    0000000000000000   0000000000000000   0000000000000000   0000000000000000
26    0000000000000000   0000000000000000   0000000000000000   0000000000000000
27    0000000000000000   0000000000000000   0000000000000000   0000000000000000
28    0000000000000000   0000000000000000   0000000000000000   0000000000000000
29    0000000000000000   0000000000000000   0000000000000000   0000000000000000
30    0000000000000000   0000000000000000   0000000000000000   0000000000000000
31    0000000000000000   0000000000000000   0000000000000000   0000000000000000

TRIGGERS
;MONEOMLocation
0X0001M        COMPLEX
; See Manual for Block Logic.
; The Flag (Reserved word) indicates a PASS Simple Trigger
;Block   Exp.    Mask    Value   Count   Goto1   Goto2   Goto3   Flag
0        0X2001  0XFFE0  0X0820  0X0000  0X0001  0X0002  0X0001  0X0002
1        0X2005  0X0000  0X0000  0X0012  0X0002  0X0003  0X0002  0X0002
2        0X2005  0XFFFF  0XDEAD  0X0000  0X0001  0X0004  0X0001  0X0002
3        0X2000  0X0000  0X0000  0X0000  0X0004  0X0004  0X0004  0X0002
4        0X0000  0X0000  0X0000  0X0000  0X0005  0X0005  0X0005  0X0000
5        0X0000  0X0000  0X0000  0X0000  0X0006  0X0006  0X0006  0X0000
6        0X0000  0X0000  0X0000  0X0000  0X0007  0X0007  0X0007  0X0000
7        0X0000  0X0000  0X0000  0X0000  0X0008  0X0008  0X0008  0X0000
8        0X0000  0X0000  0X0000  0X0000  0X0009  0X0009  0X0009  0X0000
9        0X0000  0X0000  0X0000  0X0000  0X000A  0X000A  0X000A  0X0000
10       0X0000  0X0000  0X0000  0X0000  0X000B  0X000B  0X000B  0X0000
11       0X0000  0X0000  0X0000  0X0000  0X000C  0X000C  0X000C  0X0000
12       0X0000  0X0000  0X0000  0X0000  0X000D  0X000D  0X000D  0X0000
13       0X0000  0X0000  0X0000  0X0000  0X000E  0X000E  0X000E  0X0000
14       0X0000  0X0000  0X0000  0X0000  0X000F  0X000F  0X000F  0X0000
15       0X0000  0X0000  0X0000  0X0000  0X0010  0X0010  0X0010  0X0000

END
```

C: Dynamic Link Library Calls

Most features of the PASS-1000 may be accessed from external programs through the use of Dynamic Link Library (DLL) calls. These features include hardware initialization and shutdown, BC and RT functions, monitoring, and data logging. This appendix details the DLL calls which control each of these functions.

Two DLL header files are included with the distribution of the PASS-1000 software. The PASSLIB.H header file is intended for use with a single PASS device. The PASSLIB2.H header file is for use with multiple PASS devices (maximum of eight). **Be sure that you select the header file appropriate for your PASS configuration.**

The following pages detail the DLL functions found in each of the two header files.

C.1 *PASSLIB.H* Functions

C.1.1 Hardware Setup Functions

These functions initialize and close the PASS-1000 hardware. Each DLL session must call both of these functions: the appropriate initialization function (InitMonitor, InitADI3, InitSADI3, Init SA32_1, InitSA32_2, InitSNT3, InitPCI, or InitMCIA) to start and CloseMonitor to end the session.

```
void InitMonitor ( HWND ParentHWnd, WORD IO, WORD MEMBASE,
WORD INTLEVEL, WORD NOPASS);
```

Initializes the PC2 PASS-1000 hardware in either high memory or DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board.

The **`ParentHWnd`** parameter is the calling window when called from a Borland C++ application and a 0 for all other calling applications. **`I/O`** is the board's I/O port and **`MEMBASE`** is the upper word of the board's memory address. **`INTLEV-`**

EL is the board's interrupt level. **NOPASS** is a Boolean value indicating whether the PASS software is running. Some of the DLL functions, such as GetRTData, are designed to run only when the PASS is running.

```
void InitADI3 ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
INTLEVEL, WORD NOPASS, char *FWFile);
```

Initializes the PC3 PASS-1000 hardware as a large window in DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board.

```
void InitSADI3 ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
INTLEVEL, WORD NOPASS, char *FWFile);
```

Initializes the PC3 (single channel board) PASS-1000 hardware as a small window in DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board.

```
void InitSA32_1 ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
INTLEVEL, WORD NOPASS, char *FWFile);
```

Initializes Channel 1 on the PC3-2 (dual channel board) PASS-1000 hardware as a small window in DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board.

```
void InitSA32_2 ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
INTLEVEL, WORD NOPASS, char *FWFile);
```

Initializes Channel 2 on the PC3-2 (dual channel board) PASS-1000 hardware as a small window in DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board.

Note: You cannot initialize more than one device at a time using the DLL functions found in the PASSLIB.H file and it is not possible to initialize Channel 2 alone. Therefore, the InitSA32_2 DLL function is only valid if Channel 1 (on the same board) is first initialized using the PASS-1000 software.

```
void InitSNT3 ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
IN TLEVEL, WORD NOPASS, char *FWFile);
```

Initializes the PC3 (single channel board) PASS-1000 hardware as a small window in DOS memory when the board is installed in a system that is running

Windows NT. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board.

```
void InitPCI ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
INTLEVEL, WORD NOPASS, char *FWFile);
```

Initializes the PCI PASS-1000 hardware. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board.

Note: `FWFile` is the firmware file name, *FWPC3.TXT*. All of the remaining parameters are the same as those for InitMonitor.

```
void InitMCIA ( HWND ParentHWnd, char * devname, WORD NOPASS
);
```

Initializes the PCMCIA PASS-1000 hardware as a large window in DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board.

`devname` is the name of the BRM device from the *pass.cgf* file and specifies which of the two possible pcmcia devices is to be initialized. The ParentHWnd and NOPASS parameters are the same as those for InitMonitor.

```
WORD GetWinVer( void );
```

This function returns the current version of Windows. It is used to determine whether PCI cards are mapped directly or through the **.vxd* file.

```
BOOL MapPCIW95(WORD device );
```

This function maps a single PCI device via the *Win95.vxd* file (i.e., gets selectors for memory and interrupt). It returns "true" if the mapping was successful.

```
BOOL UnMapPCIW95(WORD device );
```

This function unmaps by freeing the selectors for a single PCI device via the *Win95.vxd* file. It returns "true" if the unmapping was successful.

```
DWORD GetW95PCIAdr( WORD SBSIndex );
```

This function returns the base address of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
DWORD GetPCIMonLinAdr( WORD SBSIndex );
```

This function returns the Monitor Linear Address of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
DWORD GetPCIBRLinAdr( WORD SBSIndex );
```

This function returns the BR Linear Address of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
WORD GetPCIMonSel( WORD SBSIndex );
```

This function returns the selector for the monitor part of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
WORD GetPCIBRMSel( WORD SBSIndex );
```

This function returns the selector for the BRM part of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
WORD GetPCICSRSel( WORD SBSIndex );
```

This function returns the selector for the CSR of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
WORD GetPCIRegSel( WORD SBSIndex );
```

This function returns the selector for the PLX Bus Registers of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
WORD GetW95Irq( WORD SBSIndex );
```

This function returns the interrupt of the PCI device numbered SBSIndex. The interrupt is FFh if Win95/PCI plug-and-play did not have a free interrupt to allocate.

```
BOOL MapPCIDevices( void );
```

This function finds all PCI devices, computes a lowest address for all non-SBS devices and remaps all SBS devices starting at an address 100000000h bytes below the lowest address. It then builds an internal data structure, saving the new addresses. This mapping function should only be used in Windows 3.1 or Windows 3.11.

```
DWORD GetPCIBaseAdr( WORD DevIndex );
```

This function returns the base address of the SBS PCI device with index DevIndex. It retrieves the address from the data structure built by **`MapPCIDevices`** and, as such, **`MapPCIDevices`** must be called prior to **`GetPCIBaseAdr`**.

```
DWORD GetPCIRegAdr( WORD DevIndex );
```

This function returns the register address of the SBS PCI device with index DevIndex. It retrieves the address from the data structure built by **`MapPCIDevices`** and, as such, **`MapPCIDevices`** must be called prior to **`GetPCIBaseAdr`**.

```
WORD GetPCIIRQ( WORD DevIndex );
```

This function returns the interrupt line assigned to a given SBS PCI device. The PCI devices need to be mapped first using **`MapPCIDevices`**.

```
void CloseClient( void );
```

This function closes the PCMCIA Client application. If a PCMCIA board has been initialized without the PASS, call this function before the **`CloseMonitor`** function to insure that the client is closed before the shared global memory is released.

```
void CloseMonitor( void );
```

This function releases the memory associated with the DLL objects and frees the DLL. This function must be called when exiting a program that called any of the initialization functions.

C.1.2 Hardware Control Functions

```
BOOL StartUC( void );
```

This function causes the board's firmware to start processing.

```
void StopUC( void );
```

This function causes the board's firmware to stop processing.

C.1.3 Data Logging Functions

These functions control data logging when the PASS is not running. Logging is controlled by four calls.

```
void MonitorAll( void );
```

Sets up the PASS filter table to monitor all RTs and SAs.

```
void InitializeLog( int SAVEFILE, long int FILESIZE );
```

Sets up the Interrupt handler. SAVEFILE is the integer file handle returned by the Windows OpenFile call and FILESIZE is the maximum number of bytes the archive file can use. Once started, data logging will run until it comes to within one full buffer of the specified FILESIZE.

```
void StartMonitor( void );
```

Starts the data logging.

```
void StopMonitor( void );
```

Stops the data logging before the `FILESIZE` parameter is reached.

C.1.4 Data Log File Conversion Functions

These functions convert a PASS save file to another file format.

```
BOOL ConvertBA( char *InputFile, char *OutputFile );
```

This function converts a binary log file to an ASCII file for editing.

Warning: This function should be used with extreme caution. The ASCII file will be more than ten times the size of the binary file. Any changes made to the ASCII file must preserve the size of the binary equivalent if the file is to be converted back to a binary file for replay or display. Changes to the ASCII file should be limited to editing data values or the RT-SA of messages. Messages cannot be added or deleted and word counts cannot be altered.

```
WORD ConvertAB( char * InputFile, char *OutputFile );
```

This function converts the ASCII log files created by ConvertBA back to the PASS binary format. See the Warning box above for details.

```
WORD Convert23( char * InputFile, char *OutputFile );
```

This function converts the save files created by the PC2 version of the PASS (***.BAM*** format) to the format used by the PC3/PCMCIA version of the PASS

(***.BA3** format). These save files are created using the "Save Current Device Setup as File" option in the PASS software File menu. The PC3/PCMCIA version has 0x800 fewer words of memory than does the PC2 version; as such, setups defined by a PC2 PASS board may not fit on a PC3 PASS board. If the setup file does not fit, SIZE_ERR will be returned and the output file will be erased.

```
WORD Convert32( char * InputFile, char *OutputFile );
```

This function converts the save files created by the PC3/PCMCIA version of the PASS (***.BA3** format) to the format used by the PC2 version of the PASS (***.BAM** format). See the description of the Convert23 function.

C.1.5 Snapshot Monitor Functions

These functions start and stop the PASS Snapshot monitor. They are independent of the PASS.

```
void StartSnapshot( WORD tg_loc, WORD err_only, WORD
mon_spurious );
```

The parameters of this function specify trigger location and the filtering of errors and spurious data during snapshot monitoring. The trigger location (`trig_loc`) is "0" for a trigger at the start of the buffer, "1" for a trigger at the middle, and "2" for a trigger at the end of the buffer. `Err_only` and `mon_spurious` are boolean values. A "True" `err_only` value results in only messages with errors being stored in the snapshot buffer. A "True" `mon_spurious` value prevents spurious data from being stored. The default values are err_only = false and mon_spurious = true.

```
void StopSnapshot( void );
```

This function stops the PASS Snapshot monitor before the buffer is full.

C.1.6 BC Functions

These functions load and run the ASCII format BC list. They may be used with or without the PASS running. If the PASS is running, follow these precautions: Any BC list loaded by these functions will not be displayed by the PASS BC list dialog. The PASS BC list and RT define functions use a different memory allocation scheme than does the DLL function and should not be used once a BC list is loaded by the DLL.

```
void OpenBCFile( char * filename, WORD CLK_CONST );
```

Creates the BC file handling object, opens the BC file, and saves the board clock constant (either 15 or 14 for the MIPS rating of the mother board). This function should be called first before loading or running a BC file.

```
WORD Load_BC( WORD Forever, long int Icount );
```

Formats the ASCII BC file into PASS BC list format and loads it into board memory. Returns a flag indicating success or the reason for failure. A "true" Forever flag indicates the BC list is to run continuously. A "false" Forever flag with a nonzero Icount indicates the BC list is to be run a fixed number of times. A "false" Forever flag and a zero Icount indicates the List is to be run one time.

The **`Load_BC`** function reads the starting address for the BC chain from location 0xFFFF on the PASS mother board. If **`InitMonitor`** is called with the **`NoPass`** flag set, this location will be set to 0x1440 which is the first available memory location on the mother board. Calling **`Load_RT`** will advance this location as RT buffers are defined. Any RT setup file must, therefore, be loaded before the BC list.

```
WORD ReadBCMess( WORD * message );
```

Reads the next message from an open BC list file and returns the message in a 180 word buffer pointed to by message. This function is called by both the PASS software and **`Load_BC`** to read the BC file. An example of its use in verifying a BC list file is included in *PASSTEST.CPP*.

```
WORD Start_BC( void );
```

Starts a loaded BC list. **`OpenBCFile`** and **`Load_BC`** must be run before the list is started.

```
void Stop_BC( void );
```

Stops a running BC list.

```
void CloseBCFile( void );
```

Closes the BC list file and deletes the file object created in **`OpenBCFile`**. This function must be called to free the memory associated with the BC object.

C.1.7 BC Edit Functions

These functions allow data words in a BC-RT message to be edited while the BC list is running. The functions require that either the PASS or the DLL is currently running a BC list. For the DLL, this requires that the following functions have been called:

- **`InitMonitor`**
- **`OpenBCFile`**
- **`Start_BC`**

The BC Edit function requires the following calls, in the order shown:

- **`BC_Start_Edit`**
- **`BC_Find_Message`**
- **`BC_Find_Buffer`**
- **`BC_Get_Buffer (if you need to use or verify the current data buffer)`**
- **`Convert_BC_Buffer or Convert_Buffer_NoErr`**
- **`Put_BC_Buffer`**
- **`BC_End_Edit`**

```
WORD BC_Start_Edit( void );
```

Initializes an object to handle the editing. Returns "1" if the board has been initialized and the BC list is running; returns "0" otherwise.

```
WORD BC_Find_message( WORD Msg, WORD *BLKADR, WORD *OFFSET );
```

Finds the address on the board of message number **`Msg`** and returns the address of the header word and the offset to the value of the first word in the message. The user is responsible for computing the number of the message. Frame Syncs and Wait for External Triggers are counted as messages. The message count is one indexed. Only BC-RT messages have data buffers in the BC list and are the only messages that can be edited by this function. The function will return "1" if it finds the message, "0" otherwise.

```
WORD BC_Find_Buffer( WORD Buf, WORD msgadr, WORD msgofs, WORD
*BLKADR, WORD *OFFSET );
```

Finds the address of the header word and the offset of the first data word of the data buffer specified by **`Buf`** and returns the value in **`BLKADR`** and **`OFFSET`**. The block address and offset of the beginning of the message returned by BC_Find_Message needs to be provided in **`msgadr`** and **`msgofs`**. Each BC-RT message can have several data buffers. This function finds the specified buffer for editing. The data buffers are one indexed. The user is required to know which buffer is to be edited. The function returns "1" is successful, "0" otherwise.

```
WORD Get_BC_Buffer( WORD BLKADR, WORD OFFSET, WORD *Buffer,
WORD *
WRDCNT );
```

This function retrieves the data buffer found at board location `BLKADR`, `OFFSET` and copies it to a 100-word long buffer pointed to by `Buffer`. The data is converted to all long word format with two error words and one data word for each word. The number of valid data words is returned in `WRDCNT`. The function returns "1" if no errors were found in the copying process; "0" otherwise.

```
WORD Convert_BC_Buffer( WORD *InBuffer, WORD *OutBuffer, ER-
RPTR errors, WORD buf_size );
```

This routine converts a 33-word data array and a linked list of error entries to a PASS format BC data buffer. The PASS saves all data in long word format with two error words for each data word until the data is written to the board. The error words in the ERR structure use the PASS board bit error codes (see Appendix A) for all non-standard entries except a Gap before a particular data field. To produce this error, the MT_Gap flag second nibble used in the PASS can be ORed with the first error word. MT_Gap is defined in BCDEFS.H. The ERR structure is defined in BCEDIT.H and consists of the number of the word on which the error occurred, the two error words, and a pointer to the next error. A NULL pointer is used to indicate no errors or the end of the error list. The word number is zero indexed to follow the C array convention.

```
WORD Convert_Buffer_NoErr( WORD *InBuffer, WORD *OutBuffer,
WORD buf_size );
```

This version of `Convert_Buffer` was written for use in LabVIEW for Windows which cannot create linked lists. This function converts up to 32 data words to the PASS format but cannot inject any errors.

```
WORD Put_BC_Buffer( WORD BLKADR, WORD OFFSET, WORD *Buffer );
```

This function replaces the buffer starting at `BLKADR`, `OFFSET` with the PASS format buffer pointed to by `Buffer`. It first verifies that the two buffers are the same size. If the buffers are not the same size, the functions returns "0"; otherwise, the data words are replaced and the function returns "1". The user can loop on the BCCPTR (0x85) on the PASS mother board and load the new buffer after the address of the edited message has been processed.

```
WORD Count_BC_Buffer( WORD BLKADR, WORD OFFSET );
WORD Count_Edit_Buffer( WORD *Buffer );
```

These two functions are used by `Put_BC_Buffer` to verify the sizes of the buffer on the board and the memory array. `Count_BC_Buffer` returns the number of words used by the buffer starting at the given address and offset. `Count_Edit_Buffer` counts the number of words required to place the PASS format buffer pointed to by Buffer on the board. These functions can be used to determine how much space is available on the board for a given buffer and how much space the new buffer will require when converted to BC-List format.

```
void BC_End_Edit( void );
```

This function closes the C++ object used to handle the edit and releases all memory and resources used. It must be called at the end of the edit.

C.1.8 RT Functions

These functions load an RT setup when the PASS is not running and retrieve RT data when the PASS is running.

The first three functions are used to load an ASCII RT setup file. They are called in the following order:

- **`OpenRTFile`**
- **`LoadRTSetup`**
- **`CloseRTFile`**

```
void OpenRTFile( char * filename, WORD CLK_CONST );
```

Creates the RT file handling object, opens the RT file, and saves the board clock constant (either 15 or 14 for the MIPS rating of the mother board). This function should be called first before loading an RT file.

```
WORD LoadRTSetup( void );
```

This function calls GetNextRTDEF and GetNextSABuffer to load an RT setup file.

```
void CloseRTFile( void );
```

This function closes the RT setup file and deletes the file handling object. It must be called at the end of the RT load procedure.

```
WORD SetupRtSa( WORD RT, WORD SA, WORD Transmit );
```

This function sets up the three board buffers needed for real-time RT/SA data retrieval. It returns the number of buffers added (0, 1, 2, or 3) or an Error Message if unable to allocate the buffers. The possible return errors are as follows:

FFFFh	the requested RT is an emulated transmit RT
FFFEh	there was insufficient board memory for the required data buffers

If the call is successful, the returned buffer count must be saved to be used as a parameter on a DeleteRtSa function call when the RT data is no longer required.

```
WORD DeleteRtSa( WORD RT, WORD SA, WORD Transmit, WORD
Delete_Cnt );
```

This function is used in conjunction with SetupRtSa to allocate and deallocate RT-SA buffers independent of the PASS software. It frees the memory allocated outside of the PASS software.

```
WORD GetNextRTDEF( WORD * RT, WORD * Def, char * Name );
```

This function returns the next RT definition table entry from an open RT file. The RT number is returned to the word pointed to by `RT`. The RT label is copied to an 11-character buffer pointed to by `Name`. `Def` points to a 65-word buffer which corresponds to the PASS RT definition table entry.

```
WORD GetNextSABuffer( WORD * SA, WORD * SABUF );
```

This function returns the next subaddress and the associated header information and data buffer from an open RT file. It is called in a loop after a successful `GetNextRTDef` call to retrieve all the subaddresses for the RT. The function returns flags to indicate errors, the end of the file, or the last subaddress for the RT.

An example of reading an RT setup file using both `GetNextRTDEF` and `GetNextSABuffer` is found in *passtest.cpp*.

```
void GetRtData( WORD RT, WORD SA, WORD TR, WORD *DATA );
```

Retrieves the contents of an RT data buffer from a mapped monitored transmit RT or a mapped monitored or emulated receive RT. This function is designed to retrieve data from a running PASS.

```
void PutRtData( WORD RT, WORD SA, WORD * DATA );
```

Puts data into an RT Transmit Data Buffer. This function is designed to supply data to a running PASS. Note: New data is only written to the data buffer if the existing data in the buffer has already been transmitted on the bus. The function returns TRUE (1) if the buffer was updated and FALSE (0) if it was not updated.

```
void ForcePutRtData( WORD RT, WORD SA, WORD * DATA );
```

Puts data into an RT Transmit Data Buffer. This function is designed to supply data to a running PASS. Note: This function always updates the data buffer, regardless of whether or not the existing data has been transmitted on the bus.

C.1.9 Tool Functions

The Tool functions allow an ASCII Miscellaneous file from the PASS software to be interpreted or loaded onto the PASS board. These functions should not be used when the PASS software is running.

The first three functions load the ASCII tool file. They are called in the following order:

- ➢ **`OpenToolFile`**
- ➢ **`LoadToolFile`**
- ➢ **`CloseToolFile`**

```
void OpenToolFile( char *fname, WORD BRM_CLK_CONST, WORD
MFM_CLK_CONST );
```

This function creates the C++ object that handles the file read, opens the file names in **`fname`**, and saves the two clock constants for use by **`TGetTimeOut`**.

```
WORD LoadToolFile( char *fname, WORD BRM_CLK_CONST, WORD
MFM_CLK_CONST )
```

This function will open the file named in **`fname`**, and load the output level, response timeout values, filter table, and defined triggers onto the PASS board. The IRIG setup and the 1553A mode names and Transmit/Receive value are only used by the PASS software and are discarded. The parameters **`BRM_CLK_CONST`** and **`MFM_CLK_CONST`** are used to convert the response timeout value to board values. The BRM value would be either 14 or 15; the MFM value is 10. **`LoadToolFile`** returns "1" is the load is successful; "0" otherwise.

`LoadToolFile` uses the following functions to interpret the Miscellaneous file. The user can also use these functions to verify the read of the file. An example of the use of these functions can be found in ***passtest.cpp***.

```
void CloseToolFile( void );
```

Closes file **`fname`** and deletes the object created in **`OpenToolFile`**.

```
WORD TGetNextBlankLine( void );
```

The individual sections of the Miscellaneous file are separated by blank lines. If an error occurs in reading one section, this function can be called to read past the unreadable lines to the end of the section for error recovery.

```
WORD TGetNextType( WORD *type );
```

This function returns the type of the next section of the Miscellaneous file. The type words are defined in ***TOOLREAD.H***. If an error occurs, it will return one of the error values defined in ***TOOLREAD.H***. Once the type of the next section has been determined, a Switch statement should be used to call the appropriate function from the following functions to read the next section.

```
WORD TGetOutLevel( WORD *level );
```

This reads the output level from the file and places the word corresponding to the PASS board output level in **`level`**. The return word will either be **`READ_OK`** or will specify the error that was encountered.

```
WORD TGetTimeOut( WORD *brm_timeout, WORD *mfm_timeout );
```

Returns the board word value equivalent for the maximum Status Response time allowed before an error is reported.

```
WORD TGetModes( char *NAMES, WORD *tr );
```

This function retrieves the names and transmit/receive values for any Mode codes generated by a BC list or received by an RT defined as using the 1553A protocol. The function fills the Names buffer, defined as char TNames[32][13], with any mode code names saved in the file. The **`char`** * value parameter can be produced by taking the address of the first entry (&(Names[0][0])). The **`tr`** parameter points to an array of 32 words which will determine the value of the T/R bit for any mode code sent to an RT specified as ***1553A*** by a PASS BC list. A "0" value specifies receive and a "1" specifies transmit. These values are discarded when the Miscellaneous file is loaded from the DLL.

```
WORD TGetIrig( WORD *type, WORD *base, int *freq, long int
*offset );
```

This function retrieves the values used by the PASS software to access an IRIG card. The DLL does not currently support the use of IRIG cards and these values are not used when the file is loaded by the DLL.

```
WORD TGetFilters( WORD *ERR_ONLY, WORD *FILTERS );
```

The filters are returned in an array defined as WORD TFILTERS[32][4] with four filter words for each RT. The address of the first entry should be passed as the WORD pointer. The **`ERR_ONLY`** value is a boolean value indicating whether or not the PASS stores only error containing messages.

```
WORD TGetTriggers( WORD *noeom, WORD *location, WORD *simple,
WORD *TRIGGERS );
```

The trigger blocks are returned in an array defined as WORDTTRIGGERS[16][8]. The PASS board uses 16 blocks of 8 words each to define the trigger logic. This logic is explained in the PASS manual. The address of the first entry should be passed as the word pointer parameter. The **`noeom`** value determines which block the logic returns to on an end of message. The **`location`** determines where in the stored data the trigger will be located: beginning, middle, and end. **`Simple`** is a boolean value which designates the trigger as one of the PASS software's predefined triggers.

C.1.10 Utility Functions

These miscellaneous functions may be used with or without the PASS software.

```
WORD MGR( WORD adr );
```

Get Ram function for retrieving data from the PASS board. This function returns the contents of a memory location from the monitor area of the board.

```
void MPR( WORD adr, WORD val );
```

Put Ram function for writing data to the PASS board. This function writes a value to a memory location in the monitor area of the board.

```
WORD BGR( WORD adr );
```

Get Ram function for retrieving data from the PASS board. This function returns the contents of a memory location from the BC-RT area of the board.

```
void BPR( WORD adr, WORD val );
```

Put Ram function for writing data to the PASS board. This function writes a value to a memory location in the BC-RT area of the board.

```
void SetMonitorClock( WORD day, WORD hour, WORD min, WORD sec
);
```

This function, for the PC2 version of the PASS, sets the PASS board's internal clock to the specified value and can be used to synchronize the message time stamps in the data log file to an external source.

```
void SetPC3Clock( WORD day, WORD hour, WORD min, WORD sec );
```

This function, for the PC3 and PCMCIA versions of the PASS, sets the PASS board's internal clock to the specified value and can be used to synchronize the message time stamps in the data log file to an external source.

C.2 *PASSLIB2.H* Functions

When using this header file, the board initialization routines return a board index. This index is then required as the last parameter for all the remaining functions that require board access.

C.2.1 Hardware Setup Functions

These functions initialize and close the PASS-1000 hardware. Each DLL session must call both of these functions: the appropriate initialization function (InitMonitor, InitADI3, InitSADI3, Init SA32_1, InitSA32_2, or InitMCIA) to start and CloseMonitor to end the session.

```
WORD InitMonitor ( HWND ParentHWnd, WORD IO, WORD MEMBASE,
WORD INTLEVEL, WORD NOPASS );
```

Initializes the PC2 PASS-1000 hardware in either high memory or DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board. The returned value is the Device Index (0-7).

The **`ParentHWnd`** parameter is the calling window when called from a Borland C++ application and a 0 for all other calling applications. **`I/O`** is the board's I/O port and **`MEMBASE`** is the upper word of the board's memory address. **`INTLEVEL`** is the board's interrupt level. **`NOPASS`** is a Boolean value indicating whether the PASS software is running. Some of the DLL functions, such as GetRTData, are designed to run only when the PASS is running.

```
WORD InitADI3 ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
INTLEVEL, WORD NOPASS, char *FWFile );
```

Initializes the PC3 PASS-1000 hardware as a large window in DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board. The returned value is the Device Index (0-7).

`*FWFile` is the firmware file name, ***FWPC3.TXT***. All of the remaining parameters are the same as those for InitMonitor.

```
WORD InitSADI3 ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
INTLEVEL, WORD NOPASS, char *FWFile );
```

Initializes the PC3 (single channel board) PASS-1000 hardware as a small window in DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board. The returned value is the Device Index (0-7).

`*FWFile` is the firmware file name, ***FWPC3.TXT***. All of the remaining parameters are the same as those for InitMonitor.

```
WORD InitSA32_1 ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
INTLEVEL, WORD NOPASS, char *FWFile );
```

Initializes Channel 1 on the PC3-2 (dual channel board) PASS-1000 hardware as a small window in DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board. The returned value is the Device Index (0-7).

`*FWFile` is the firmware file name, FWPC3.TXT. All of the remaining parameters are the same as those for InitMonitor.

```
WORD InitSA32_2 ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
INTLEVEL, WORD NOPASS, char *FWFile );
```

Initializes Channel 2 on the PC3-2 (dual channel board) PASS-1000 hardware as a small window in DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board. The returned value is the Device Index (0-7).

`*FWFile` is the firmware file name, *FWPC3.TXT*. All of the remaining parameters are the same as those for InitMonitor.

Note: It is not possible to initialize Channel 2 alone. Therefore, the `InitSA32_2 DLL` function is only valid if Channel 1 (on the same board) is first initialized using the `InitSA32_1` DLL function or the PASS-1000 software

```
WORD InitPCI ( HWND ParentHWnd, WORD IO, WORD MEMBASE, WORD
INTLEVEL, WORD NOPASS, char *FWFile );
```

Initializes the PCI PASS-1000 hardware. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board. The returned value is the Device Index (0-7).

`*FWFile` is the firmware file name, ***FWPC3.TXT***. All of the remaining parameters are the same as those for InitMonitor.

```
WORD InitMCIA ( HWND ParentHWnd, char * devname, WORD NOPASS
);
```

Initializes the PCMCIA PASS-1000 hardware as a large window in DOS memory. This function creates the low level device objects used for all board related functions. It must be run before any of the other DLL functions that access the board. The returned value is the Device Index (0-7).

`devname` is the name of the BRM device from the ***PASS.CFG*** file and specifies which of the two possible pcmcia devices is to be initialized. The ParentHWnd and NOPASS parameters are the same as those for InitMonitor.

```
WORD GetWinVer( void );
```

This function returns the current version of Windows. It is used to determine whether PCI cards are mapped directly or through the **.vxd* file.

```
BOOL MapPCIW95(WORD device );
```

This function maps a single PCI device via the *Win95.vxd* file (i.e., gets selectors for memory and interrupt). It returns "true" if the mapping was successful.

```
BOOL UnMapPCIW95(WORD device );
```

This function unmaps by freeing the selectors for a single PCI device via the *Win95.vxd* file. It returns "true" if the unmapping was successful.

```
DWORD GetW95PCIAdr( WORD SBSIndex );
```

This function returns the base address of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
DWORD GetPCIMonLinAdr( WORD SBSIndex );
```

This function returns the Monitor Linear Address of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
DWORD GetPCIBRLinAdr( WORD SBSIndex );
```

This function returns the BR Linear Address of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
WORD GetPCIMonSel( WORD SBSIndex );
```

This function returns the selector for the monitor part of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
WORD GetPCIBRMSel( WORD SBSIndex );
```

This function returns the selector for the BRM part of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
WORD GetPCICSRSel( WORD SBSIndex );
```

This function returns the selector for the CSR of the PCI device numbered SB-SIndex. It returns a zero if the board is not mapped or the mapping failed.

```
WORD GetPCIRegSel( WORD SBSIndex );
```

This function returns the selector for the PLX Bus Registers of the PCI device numbered SBSIndex. It returns a zero if the board is not mapped or the mapping failed.

```
WORD GetW95Irq( WORD SBSIndex );
```

This function returns the interrupt of the PCI device numbered SBSIndex. The interrupt is FFh if Win95/PCI plug-and-play did not have a free interrupt to al-locate.

```
WORD MapPCIDevices( void );
```

This function finds all PCI devices, computes a lowest address for all non-SBS devices and remaps all SBS devices starting at an address 100000000h bytes be-low the lowest address. It then builds an internal data structure, saving the new addresses. This mapping function should only be used in Windows 3.1 or Win-dows 3.11.

```
DWORD GetPCIBaseAdr( WORD DevIndex );
```

This function returns the base address of the SBS PCI device with index DevIndex. It retrieves the address from the data structure built by **`MapPCIDe-vices`** and, as such, **`MapPCIDevices`** must be called prior to **`GetPCIBaseAdr`**.

```
DWORD GetPCIRegAdr( WORD DevIndex );
```

This function returns the register address of the SBS PCI device with index DevIndex. It retrieves the address from the data structure built by **`MapPCIDe-vices`** and, as such, **`MapPCIDevices`** must be called prior to **`GetPCIBaseAdr`**.

```
WORD GetPCIIRQ( WORD DevIndex );
```

This function returns the interrupt line assigned to a given SBS PCI device. The PCI devices need to be mapped first using **`MapPCIDevices`**.

```
void CloseClient( void );
```

This function closes the PCMCIA Client application. If a PCMCIA board has been initialized without the PASS, call this function before the **`CloseMonitor`** function to insure that the client is closed before the shared global memory is released.

```
void CloseMonitor( void );
```

This function releases the memory associated with the DLL objects and frees the DLL. This function must be called when exiting a program that called any of the initialization functions.

C.2.2 Hardware Control Functions

```
BOOL StartUC( WORD ndx );
```

This function causes the board's firmware to start processing.

```
void StopUC( WORD ndx );
```

This function causes the board's firmware to stop processing.

C.2.3 Data Logging Functions

These functions control data logging when the PASS is not running. Logging is controlled by four calls.

```
void MonitorAll( void );
```

Sets up the PASS filter table to monitor all RTs and SAs.

```
void InitializeLog( int SAVEFILE, long int FILESIZE );
```

Sets up the Interrupt handler. **`SAVEFILE`** is the integer file handle returned by the Windows OpenFile call and **`FILESIZE`** is the maximum number of bytes the archive file can use. Once started, data logging will run until it comes to within one full buffer of the specified **`FILESIZE`**.

```
void StartMonitor( void );
```

Starts the data logging.

```
void StopMonitor( void );
```

Stops the data logging before the **`FILESIZE`** parameter is reached.

C.2.4 Data Log File Conversion Functions

These functions convert a PASS save file to another file format.

```
BOOL ConvertBA( char * InputFile, char *OutputFile );
```

This function converts a binary log file to an ASCII file for editing.

```
WORD ConvertAB( char * InputFile, char *OutputFile );
```

This function converts the ASCII log files created by ConvertBA back to the PASS binary format. See the Warning box above for details.

```
WORD Convert23( char * InputFile, char *OutputFile );
```

This function converts the save files created by the PC2 version of the PASS (***.BAM** format) to the format used by the PC3/PCMCIA version of the PASS (***.BA3** format). These save files are created using the "Save Current Device Setup as File" option in the PASS software File menu. The PC3/PCMCIA version has 0x800 fewer words of memory than does the PC2 version; as such, setups defined by a PC2 PASS board may not fit on a PC3 PASS board. If the setup file does not fit, **`SIZE_ERR`** will be returned and the output file will be erased.

```
WORD Convert32( char * InputFile, char *OutputFile );
```

This function converts the save files created by the PC3/PCMCIA version of the PASS (***.BA3** format) to the format used by the PC2 version of the PASS (***.BAM** format). See the description of the Convert23 function.

C.2.5 Snapshot Monitor Functions

These functions start and stop the PASS Snapshot monitor. They are independent of the PASS.

```
void StartSnapshot( WORD tg_loc, WORD err_only, WORD
mon_spurious,
 WORD ndx );
```

The parameters of this function specify trigger location and the filtering of errors and spurious data during snapshot monitoring. The trigger location (**`trig_loc`**) is "0" for a trigger at the start of the buffer, "1" for a trigger at the middle, and "2" for a trigger at the end of the buffer. **`Err_only`** and **`mon_spurious`** are boolean values. A "True" **`err_only`** value results in only messages with errors being stored in the snapshot buffer. A "True" **`mon_spurious`** value prevents spurious data from being stored. The default values are err_only = false and mon_spurious = true.

```
void StopSnapshot( WORD ndx );
```

This function stops the PASS Snapshot monitor before the buffer is full.

C.2.6 BC Functions

These functions load and run the ASCII format BC list. They may be used with or without the PASS running. If the PASS is running, follow these precautions: Any BC list loaded by these functions will not be displayed by the PASS BC list dialog. The PASS BC list and RT define functions use a different memory allocation scheme than does the DLL function and should not be used once a BC list is loaded by the DLL.

```
void OpenBCFile( char * filename, WORD CLK_CONST, WORD ndx );
```

Creates the BC file handling object, opens the BC file, and saves the board clock constant (either 15 or 14 for the MIPS rating of the mother board). This function should be called first before loading or running a BC file.

```
WORD Load_BC( WORD Forever, long int Icount, WORD ndx );
```

Formats the ASCII BC file into PASS BC list format and loads it into board memory. Returns a flag indicating success or the reason for failure. A "true" **`Forever`** flag indicates the BC list is to run continuously. A "false" **`Forever`** flag with a nonzero **`Icount`** indicates the BC list is to be run a fixed number of times. A "false" **`Forever`** flag and a zero **`Icount`** indicates the List is to be run one time.

The **`Load_BC`** function reads the starting address for the BC chain from location 0xFFFF on the PASS mother board. If **`InitMonitor`** is called with the **`NoPass`** flag set, this location will be set to 0x1440 which is the first available memory location on the mother board. Calling **`Load_RT`** will advance this location as RT buffers are defined. Any RT setup file must, therefore, be loaded before the BC list.

```
WORD ReadBCMess( WORD * message, WORD ndx );
```

Reads the next message from an open BC list file and returns the message in a

180 word buffer pointed to by message. This function is called by both the PASS software and **Load_BC** to read the BC file. An example of its use in verifying a BC list file is included in *PASSTEST.CPP*.

```
WORD Start_BC( WORD ndx );
```

Starts a loaded BC list. **OpenBCFile** and **Load_BC** must be run before the list is started.

```
void Stop_BC( WORD ndx );
```

Stops a running BC list.

```
void CloseBCFile( WORD ndx );
```

Closes the BC list file and deletes the file object created in **OpenBCFile**. This function must be called to free the memory associated with the BC object.

C.2.7 BC Edit Functions

These functions allow data words in a BC-RT message to be edited while the BC list is running. The functions require that either the PASS or the DLL is currently running a BC list. For the DLL, this requires that the following functions have been called:

- **InitMonitor**
- **OpenBCFile**
- **Start_BC**

The BC Edit function requires the following calls, in the order shown:

- **BC_Start_Edit**
- **BC_Find_Message**
- **BC_Find_Buffer**
- **BC_Get_Buffer (if you need to use or verify the current data buffer)**
- **Convert_BC_Buffer or Convert_Buffer_NoErr**
- **Put_BC_Buffer**
- **BC_End_Edit**

```
WORD BC_Start_Edit( WORD ndx );
```

Initializes an object to handle the editing. Returns "1" if the board has been initialized and the BC list is running; returns "0" otherwise.

```
WORD BC_Find_message( WORD Msg, WORD *BLKADR, WORD *OFFSET,
WORD ndx );
```

Finds the address on the board of message number **Msg** and returns the address of the header word and the offset to the value of the first word in the message. The user is responsible for computing the number of the message. Frame Syncs and Wait for External Triggers are counted as messages. The message count is one indexed. Only BC-RT messages have data buffers in the BC list and are the only messages that can be edited by this function. The function will return "1" if it finds the message, "0" otherwise.

```
WORD BC_Find_Buffer( WORD Buf, WORD msgadr, WORD msgofs, WORD
*BLKADR, WORD *OFFSET, WORD ndx );
```

Finds the address of the header word and the offset of the first data word of the data buffer specified by **Buf** and returns the value in **BLKADR** and **OFFSET**. The block address and offset of the beginning of the message returned by **BC_Find_Message** needs to be provided in **msgadr** and **msgofs**. Each BC-RT message can have several data buffers. This function finds the specified buffer for editing. The data buffers are one indexed. The user is required to know which buffer is to be edited. The function returns "1" is successful, "0" otherwise.

```
WORD Get_BC_Buffer( WORD BLKADR, WORD OFFSET, WORD *Buffer,
WORD *WRDCNT, WORD ndx );
```

This function retrieves the data buffer found at board location **BLKADR**, **OFFSET** and copies it to a 100-word long buffer pointed to by **Buffer**. The data is converted to all long word format with two error words and one data word for each word. The number of valid data words is returned in **WRDCNT**. The function returns "1" if no errors were found in the copying process; "0" otherwise.

```
WORD Convert_BC_Buffer( WORD *InBuffer, WORD *OutBuffer, ER-
RPTR errors, WORD buf_size, WORD ndx );
```

This routine converts a 33-word data array and a linked list of error entries to a PASS format BC data buffer. The PASS saves all data in long word format with two error words for each data word until the data is written to the board. The error words in the ERR structure use the PASS board bit error codes (see Appendix A) for all non-standard entries except a Gap before a particular data field. To produce this error, the **MT_Gap** flag second nibble used in the PASS can be ORed with the first error word. **MT_Gap** is defined in BCDEFS.H. The ERR structure is defined in BCEDIT.H and consists of the number of the word on which the error occurred, the two error words, and a pointer to the next error. A NULL pointer is used to indicate no errors or the end of the error list. The word number is zero indexed to follow the C array convention.

```
WORD Convert_Buffer_NoErr( WORD *InBuffer, WORD *OutBuffer,
WORD buf_size, WORD ndx );
```

This version of Convert_Buffer was written for use in LabVIEW for Windows which cannot create linked lists. This function converts up to 32 data words to the PASS format but cannot inject any errors.

```
WORD Put_BC_Buffer( WORD BLKADR, WORD OFFSET, WORD *Buffer,
WORD ndx );
```

This function replaces the buffer starting at **`BLKADR`**, **`OFFSET`** with the PASS format buffer pointed to by **`Buffer`**. It first verifies that the two buffers are the same size. If the buffers are not the same size, the functions returns "0"; otherwise, the data words are replaced and the function returns "1". The user can loop on the BCCPTR (0x85) on the PASS mother board and load the new buffer after the address of the edited message has been processed.

```
WORD Count_BC_Buffer( WORD BLKADR, WORD OFFSET, WORD ndx );
WORD Count_Edit_Buffer( WORD *Buffer, WORD ndx );
```

These two functions are used by **`Put_BC_Buffer`** to verify the sizes of the buffer on the board and the memory array. **`Count_BC_Buffer`** returns the number of words used by the buffer starting at the given address and offset. **`Count_Edit_Buffer`** counts the number of words required to place the PASS format buffer pointed to by Buffer on the board. These functions can be used to determine how much space is available on the board for a given buffer and how much space the new buffer will require when converted to BC-List format. Note: The user does not need to call these functions.

```
void BC_End_Edit( WORD ndx );
```

This function closes the C++ object used to handle the edit and releases all memory and resources used. It must be called at the end of the edit.

C.2.8 RT Functions

These functions load an RT setup when the PASS is not running and retrieve RT data when the PASS is running.

The first three functions are used to load an ASCII RT setup file. They are called in the following order:

- ➢ **`OpenRTFile`**
- ➢ **`LoadRTSetup`**
- ➢ **`CloseRTFile`**

```
void OpenRTFile( char * filename, WORD CLK_CONST, WORD ndx );
```

Creates the RT file handling object, opens the RT file, and saves the board clock constant (either 15 or 14 for the MIPS rating of the mother board). This function should be called first before loading an RT file.

```
WORD LoadRTSetup( WORD ndx );
```

This function calls **`GetNextRTDEF`** and **`GetNextSABuffer`** to load an RT setup file.

```
void CloseRTFile( WORD ndx );
```

This function closes the RT setup file and deletes the file handling object. It must be called at the end of the RT load procedure.

```
WORD SetupRtSa( WORD RT, WORD SA, WORD Transmit , WORD ndx);
```

This function sets up the three board buffers needed for real-time RT/SA data retrieval. It returns the number of buffers added (0, 1, 2, or 3) or an Error Message if unable to allocate the buffers. The possible return errors are as follows:

FFFFh	the requested RT is an emulated transmit RT
FFFEh	there was insufficient board memory for the required data buffers

If the call is successful, the returned buffer count must be saved to be used as a parameter on a DeleteRtSa function call when the RT data is no longer required.

```
WORD DeleteRtSa( WORD RT, WORD SA, WORD Transmit, WORD
Delete_Cnt, WORD ndx );
```

This function is used in conjunction with **`SetupRtSa`** to allocate and deallocate RT-SA buffers independent of the PASS software. It frees the memory allocated outside of the PASS software.

```
WORD GetNextRTDEF( WORD * RT, WORD * Def, char * Name, WORD
ndx );
```

This function returns the next RT definition table entry from an open RT file. The RT number is returned to the word pointed to by **`RT`**. The RT label is copied to an 11-character buffer pointed to by **`Name`**. **`Def`** points to a 65-word buffer which corresponds to the PASS RT definition table entry.

```
WORD GetNextSABuffer( WORD * SA, WORD * SABUF, WORD ndx );
```

This function returns the next subaddress and the associated header information and data buffer from an open RT file. It is called in a loop after a successful **`GetNextRTDef`** call to retrieve all the subaddresses for the RT. The function returns flags to indicate errors, the end of the file, or the last subaddress for the RT.

An example of reading an RT setup file using both **`GetNextRTDEF`** and **`GetNextSABuffer`** is found in *PASSTEST.CPP*.

```
void GetRtData( WORD RT, WORD SA, WORD TR, WORD *DATA, WORD
ndx );
```

Retrieves the contents of an RT data buffer from a mapped monitored transmit RT or a mapped monitored or emulated receive RT. This function is designed to retrieve data from a running PASS.

```
void PutRtData( WORD RT, WORD SA, WORD * DATA, WORD ndx );
```

Puts data into an RT Transmit Data Buffer. This function is designed to supply data to a running PASS. Note: New data is only written to the data buffer if the existing data in the buffer has already been transmitted on the bus. The function returns TRUE (1) if the buffer was updated and FALSE (0) if it was not updated.

C.2.9 Tool Functions

The Tool functions allow an ASCII Miscellaneous file from the PASS software to be interpreted or loaded onto the PASS board. These functions should not be used when the PASS software is running.

The first three functions load the ASCII tool file. They are called in the following order:

- ➢ **`OpenToolFile`**
- ➢ **`LoadToolFile`**
- ➢ **`CloseToolFile`**

```
void OpenToolFile( char *fname, WORD BRM_CLK_CONST, WORD
MFM_CLK_CONST, WORD ndx );
```

This function creates the C++ object that handles the file read, opens the file names in **`fname`**, and saves the two clock constants for use by **`TGetTimeOut`**.

```
WORD LoadToolFile( char *fname, WORD BRM_CLK_CONST, WORD
MFM_CLK_CONST, WORD ndx )
```

This function will open the file named in `fname`, and load the output level, response timeout values, filter table, and defined triggers onto the PASS board. The IRIG setup and the 1553A mode names and Transmit/Receive value are only used by the PASS software and are discarded. The parameters `BRM_CLK_CONST` and `MFM_CLK_CONST` are used to convert the response timeout value to board values. The BRM value would be either 14 or 15; the MFM value is 10. `LoadToolFile` returns "1" is the load is successful; "0" otherwise.

`LoadToolFile` uses the following functions to interpret the Miscellaneous file. The user can also use these functions to verify the read of the file. An example of the use of these functions can be found in ***PASSTEST.CPP***.

```
void CloseToolFile( WORD ndx );
```

Closes file **fname** and deletes the object created in `OpenToolFile`.

```
WORD TGetNextBlankLine( WORD ndx );
```

The individual sections of the Miscellaneous file are separated by blank lines. If an error occurs in reading one section, this function can be called to read past the unreadable lines to the end of the section for error recovery.

```
WORD TGetNextType( WORD *type, WORD ndx );
```

This function returns the type of the next section of the Miscellaneous file. The type words are defined in ***TOOLREAD.H***. If an error occurs, it will return one of the error values defined in ***TOOLREAD.H***. Once the type of the next section has been determined, a Switch statement should be used to call the appropriate function from the following functions to read the next section.

```
WORD TGetOutLevel( WORD *level, WORD ndx );
```

This reads the output level from the file and places the word corresponding to the PASS board output level in `level`. The return word will either be `READ_OK` or will specify the error that was encountered.

```
WORD TGetTimeOut( WORD *brm_timeout, WORD *mfm_timeout, WORD
ndx );
```

Returns the board word value equivalent for the maximum Status Response time allowed before an error is reported.

```
WORD TGetModes( char *NAMES, WORD *tr, WORD ndx );
```

This function retrieves the names and transmit/receive values for any Mode codes generated by a BC list or received by an RT defined as using the 1553A protocol. The function fills the Names buffer, defined as char TNames[32][13], with any mode code names saved in the file. The **char *** value parameter can be produced by taking the address of the first entry (&(Names[0][0])). The **tr** parameter points to an array of 32 words which will determine the value of the T/R bit for any mode code sent to an RT specified as 1553A by a PASS BC list. A "0" value specifies receive and a "1" specifies transmit. These values are discarded when the Miscellaneous file is loaded from the DLL.

```
WORD TGetIrig( WORD *type, WORD *base, int *freq, long int
*offset, WORD ndx );
```

This function retrieves the values used by the PASS software to access an IRIG card. The DLL does not currently support the use of IRIG cards and these values are not used when the file is loaded by the DLL.

```
WORD TGetFilters( WORD *ERR_ONLY, WORD *FILTERS, WORD ndx );
```

The filters are returned in an array defined as WORD TFILTERS[32][4] with four filter words for each RT. The address of the first entry should be passed as the WORD pointer. The **ERR_ONLY** value is a boolean value indicating whether or not the PASS stores only error containing messages.

```
WORD TGetTriggers( WORD *noeom, WORD *location, WORD *simple,
WORD *TRIGGERS, WORD ndx );
```

The trigger blocks are returned in an array defined as WORDTTRIGGERS[16][8]. The PASS board uses 16 blocks of 8 words each to define the trigger logic. This logic is explained in the PASS manual. The address of the first entry should be passed as the word pointer parameter. The **noeom** value determines which block the logic returns to on an end of message. The **location** determines where in the stored data the trigger will be located: beginning, middle, and end. **Simple** is a boolean value which designates the trigger as one of the PASS software's predefined triggers.

C.2.10 Utility Functions

These miscellaneous functions may be used with or without the PASS software.

```
WORD MGR( WORD adr, WORD ndx );
```

Get Ram function for retrieving data from the PASS board. This function returns the contents of a memory location from the monitor area of the board.

```
void MPR( WORD adr, WORD val, WORD ndx );
```

Put Ram function for writing data to the PASS board. This function writes a value to a memory location in the monitor area of the board.

```
WORD BGR( WORD adr, WORD ndx );
```

Get Ram function for retrieving data from the PASS board. This function returns the contents of a memory location from the BC-RT area of the board.

```
void BPR( WORD adr, WORD val, WORD ndx );
```

Put Ram function for writing data to the PASS board. This function writes a value to a memory location in the BC-RT area of the board.

```
void SetMonitorClock( WORD day, WORD hour, WORD min, WORD sec,
WORD ndx );
```

This function, for the PC2 version of the PASS, sets the PASS board's internal clock to the specified value and can be used to synchronize the message time stamps in the data log file to an external source.

```
void SetPC3Clock( WORD day, WORD hour, WORD min, WORD sec,
WORD ndx );
```

This function, for the PC3 and PCMCIA versions of the PASS, sets the PASS board's internal clock to the specified value and can be used to synchronize the message time stamps in the data log file to an external source.

C.3 Example Program

The following C++ program demonstrates the use of the PASS DLL functions. The full source code is available upon request.

```
// OWLCVT 04/10/95 09:21:59
/*********************************************************************
*                                                                    *
*                         PASSTest.cpp                               *
*                                                                    *
*    A Simple Windows Test of the PASS-LIB.DLL                       *
*                                                                    *
*********************************************************************/



#include "owl\compat.h"
#include <owl\owlall.h>
#include <stdio.h>
#include <string.h>
#include "passcfg.h"
#include "device.h"
#include "bcdefs.h"
```

```
#include "bcread.h"
#include "rtread.h"
#include "bcedit.h"
#include "toolread.h"
#include "passdll.h"

#define CM_INIT   101
#define CM_DIS80  102
#define CM_F2040  103
#define CM_D2040  104
#define CM_CLOSE  105
#define CM_ZERO   106
#define CM_INITL  201
#define CM_STARTL 202
#define CM_STOPL  203
#define CM_CONVERT  204
#define CM_BCREAD 300
#define CM_RTRead 400
#define CN_ToolRead 701



class TPASSTest : public TApplication {
public:
  TPASSTest(LPSTR Name, HINSTANCE hInstance,
  HINSTANCE hPrevInstance, LPSTR lpCmd,
  int nCmdShow)
        : TApplication(Name, hInstance,
       hPrevInstance, lpCmd, nCmdShow) {};
virtual void InitMainWindow();
};

class TTestWindow : public TFrameWindow {
public:
BOOL fileopen;
BOOL BCLoaded;
BOOL NoPass;
BOOL BC_Running;
int testfile;
OFSTRUCT AnOfstruct;

// Function pointer variables for dynamically loading DLL. See passdll.h for
typedefs
/
*************************************************************************
***********/
PTInitMonitor     InitMonitor;
PTInitADI3        InitADI3;
PTInitSADI3       InitSADI3;
PTInitMCIA        InitMCIA;
PTCloseMonitor    CloseMonitor;
PTMonitorAll      MonitorAll;
PTSetMonitorClock SetMonitorClock;
PTSetPC3Clock     SetPC3Clock;
PTMGR             MGR;
PTMPR             MPR;
PTBGR             BGR;
PTBPR             BPR;
PTStartUC         StartUC;
PTStopUC          StopUC;
PTInitializeLog   InitializeLog;
PTStartMonitor    StartMonitor;
PTStopMonitor     StopMonitor;
PTConvertBA       ConvertBA;
PTConvertAB       ConvertAB;
PTConvert23       Convert23;
```

```
PTConvert32       Convert32;
PTGetRtData       GetRtData;
PTPutRtData       PutRtData;
PTOpenBCFile      OpenBCFile;
PTCloseBCFile     CloseBCFile;
PTReadBCMess      ReadBCMess;
PTOpenRTFile      OpenRTFile;
PTCloseRTFile     CloseRTFile;
PTGetNextRTDEF    GetNextRTDEF;
PTGetNextSABuffer GetNextSABuffer;
PTStartSnapshot   StartSnapshot;
PTStopSnapshot    StopSnapshot;
PTLoad_BC         Load_BC;
PTStart_BC        Start_BC;
PTStop_BC         Stop_BC;
PTBC_Start_Edit   BC_Start_Edit;
PTBC_End_Edit     BC_End_Edit;
PTBC_Find_Message BC_Find_Message ;
PTBC_Find_Buffer  BC_Find_Buffer;
PTGet_BC_Buffer   Get_BC_Buffer;
PTCount_BC_Buffer Count_BC_Buffer;
PTCount_Edit_Buffer  Count_Edit_Buffer;
PTPut_BC_Buffer      Put_BC_Buffer;
PTConvert_BC_Buffer  Convert_BC_Buffer;
PTLoadRTSetup        LoadRTSetup;
PTOpenToolFile       OpenToolFile;
PTCloseToolFile      CloseToolFile;
PTGetNextBlankLine   TGetNextBlankLine;
PTGetNextType        TGetNextType;
PTGetTimeOut         TGetTimeOut;
PTGetOutLevel        TGetOutLevel;
PTGetModes           TGetModes;
PTGetIrig            TGetIrig;
PTGetFilters         TGetFilters;
PTGetTriggers        TGetTriggers;
PTLoadToolFile       LoadToolFile;
HINSTANCE         PassLib;
BOOL              BadLib;
/*********************************************************************/

TTestWindow(TWindow * AParent, LPSTR ATitle);
void CMINIT();
void CMDIS80();
void CMF2040();
void CMD2040();
void CMZERO();
void CMCLOSE();
void CMINITL();
void CMSTARTL();
void CMSTOPL();
void CMCONVERTBA();
void CMCONVERTAB();
void CMCONVERT23();
void CMCONVERT32();
void CMRT1();
void CMRT1TRANS();
void CMBCLOAD();
void CMBCSTART();
void CMBCSTOP();
void CMBCCLOSE();
void CMBCREAD();
void CMBCEDIT();
void CMRTREAD();
void CMRTLOAD();
void CMSTARTSS();
void CMSTOPSS();
void CMTOOLREAD();
void CMTOOLLOAD();

DECLARE_RESPONSE_TABLE( TTestWindow );
};

TTestWindow::TTestWindow(TWindow * AParent, LPSTR ATitle)
: TFrameWindow(AParent, ATitle)
{
AssignMenu("MAINMENU");
fileopen = FALSE;
// NOPASS is used to tell the DLL if the PASS is currently running.
// For some functions, such as GetRtData. The PASS software is assumed
// to be running.

#ifdef NOPASS
NoPass = TRUE;
#else
NoPass = FALSE;
#endif
BC_Running = FALSE;
BCLoaded = FALSE;
/*******************************************************************/
// The following section acts to load the dll and get pointers to the func-
tions.
// After this, the calls are identical to the statically linked library.

BadLib = FALSE;
PassLib = LoadLibrary("passlib.dll");

// This checks for a correct load. Values below 32 are error messages.
if((int)PassLib < 32)
{
BadLib = TRUE;
::MessageBox(0,"Unable to Load passlib.lib","Error",MB_OK);
}
else
{

// GetProcAddress returns a pointer to a function. The type casts make the
// functions callable using the same names as before.

InitMonitor     = (PTInitMonitor) GetProcAddress(PassLib,"InitMonitor");
InitADI3        = (PTInitADI3) GetProcAddress(PassLib,"InitADI3");
InitSADI3       = (PTInitSADI3) GetProcAddress(PassLib,"InitSADI3");
InitMCIA        = (PTInitMCIA) GetProcAddress(PassLib,"InitMCIA");
CloseMonitor    = (PTCloseMonitor) GetProcAddress(PassLib, "CloseMonitor");
MonitorAll      = (PTMonitorAll) GetProcAddress(PassLib, "MonitorAll");
SetMonitorClock = (PTSetMonitorClock) GetProcAddress(PassLib,
"SetPC3Clock");
SetPC3Clock     = (PTSetPC3Clock) GetProcAddress(PassLib, "SetMonitor-
Clock");
MGR             = (PTMGR) GetProcAddress(PassLib, "MGR");
MPR             = (PTMPR) GetProcAddress(PassLib, "MPR");
BGR             = (PTMGR) GetProcAddress(PassLib, "BGR");
BPR             = (PTMPR) GetProcAddress(PassLib, "BPR");
StartUC         = (PTStartUC) GetProcAddress(PassLib, "StartUC");
StopUC          = (PTStopUC) GetProcAddress(PassLib, "StopUC");
InitializeLog   = (PTInitializeLog) GetProcAddress(PassLib, "InitializeL-
og");
StartMonitor    = (PTStartMonitor) GetProcAddress(PassLib, "StartMonitor");
StopMonitor     = (PTStopMonitor) GetProcAddress(PassLib, "StopMonitor");
StartSnapshot   = (PTStartSnapshot) GetProcAddress(PassLib, "StartSnap-
shot");
StopSnapshot    = (PTStopSnapshot) GetProcAddress(PassLib, "StopSnapshot");
ConvertBA       = (PTConvertBA) GetProcAddress(PassLib, "ConvertBA");
```

```
ConvertAB       = (PTConvertAB) GetProcAddress(PassLib, "ConvertAB");
Convert23       = (PTConvert23) GetProcAddress(PassLib, "Convert23");
Convert32       = (PTConvert32) GetProcAddress(PassLib, "Convert32");
GetRtData       = (PTGetRtData)  GetProcAddress(PassLib, "GetRtData");
PutRtData       = (PTPutRtData)  GetProcAddress(PassLib, "PutRtData");
OpenBCFile      = (PTOpenBCFile)GetProcAddress(PassLib, "OpenBCFile");
ReadBCMess      = (PTReadBCMess)GetProcAddress(PassLib, "ReadBCMess");
Load_BC         = (PTLoad_BC)GetProcAddress(PassLib, "Load_BC");
Start_BC        = (PTStart_BC)GetProcAddress(PassLib, "Start_BC");
Stop_BC         = (PTStop_BC)GetProcAddress(PassLib, "Stop_BC");
CloseBCFile     = (PTCloseBCFile)GetProcAddress(PassLib, "CloseBCFile");
OpenRTFile      = (PTOpenRTFile)GetProcAddress(PassLib, "OpenRTFile");
CloseRTFile     = (PTCloseRTFile)GetProcAddress(PassLib, "CloseRTFile");
GetNextRTDEF    = (PTGetNextRTDEF)GetProcAddress(PassLib, "GetNextRTDEF");
GetNextSABuffer = (PTGetNextSABuffer)GetProcAddress(PassLib, "GetNext-
SABuffer");
LoadRTSetup     = (PTLoadRTSetup)GetProcAddress(PassLib, "LoadRTSetup");
BC_Start_Edit  = (PTBC_Start_Edit)GetProcAddress(PassLib, "BC_Start_Edit");
BC_End_Edit     = ( PTBC_End_Edit)GetProcAddress(PassLib, "BC_End_Edit");
BC_Find_Message = ( PTBC_Find_Message)GetProcAddress(PassLib,
"BC_Find_Message") ;
BC_Find_Buffer  = ( PTBC_Find_Buffer)GetProcAddress(PassLib,
"BC_Find_Buffer");
Get_BC_Buffer   = ( PTGet_BC_Buffer )GetProcAddress(PassLib,
"Get_BC_Buffer");
Count_BC_Buffer = ( PTCount_BC_Buffer )GetProcAddress(PassLib,
"Count_BC_Buffer");
Count_Edit_Buffer = ( PTCount_Edit_Buffer )GetProcAddress(PassLib,
"Count_Edit_Buffer");
Put_BC_Buffer     = ( PTPut_BC_Buffer )GetProcAddress(PassLib,
"Put_BC_Buffer");
Convert_BC_Buffer = ( PTConvert_BC_Buffer )GetProcAddress(PassLib,
"Convert_BC_Buffer");
OpenToolFile =      ( PTOpenToolFile )GetProcAddress(PassLib, "OpenToolF-
ile");
LoadToolFile =      ( PTLoadToolFile )GetProcAddress(PassLib, "LoadToolF-
ile");
CloseToolFile =     ( PTCloseToolFile )GetProcAddress(PassLib, "CloseToolF-
ile");
TGetNextBlankLine = ( PTGetNextBlankLine )GetProcAddress(PassLib, "TGetNext-
BlankLine");
TGetNextType =     ( PTGetNextType )GetProcAddress(PassLib, "TGetNextType");
TGetTimeOut =       ( PTGetTimeOut )GetProcAddress(PassLib, "TGetTimeOut");
TGetOutLevel =     ( PTGetOutLevel )GetProcAddress(PassLib, "TGetOutLevel");
TGetModes =         ( PTGetModes )GetProcAddress(PassLib, "TGetModes");
TGetIrig =          ( PTGetIrig )GetProcAddress(PassLib, "TGetIrig");
TGetFilters =       ( PTGetFilters )GetProcAddress(PassLib, "TGetFilters");
TGetTriggers =     ( PTGetTriggers )GetProcAddress(PassLib, "TGetTriggers");

}
/*********************************************************************/

}

void TTestWindow::CMINIT()
{
/* This function initializes the pointers to both boards and, if NoPass is
true,
initializes the boards and starts the microcode. It needs to be called before
all functions except the Read BC File function.
*/

// Check for correct loading of the dll before calling any of the functions.
if(!BadLib)
{
// dos pc3 large memory model
```

```
#ifdef TYPEADI3
InitADI3(HWindow,IOPORT,ADDRESS,INTP,NoPass,"FWPC3.TXT");
#endif

// dos pc3 small memory model
#ifdef TYPESADI3
InitSADI3(HWindow,IOPORT,ADDRESS,INTP,NoPass,"FWPC3.TXT");
#endif

// SMALL WINDOW PC-MCIA
#ifdef TYPESMCIA
InitMCIA(HWindow,"MCIABRM",NoPass);
#endif

// pc2
#ifdef TYPEAXI
InitMonitor(HWindow,IOPORT,ADDRESS,INTP,NoPass);

if(NoPass)
StartUC();
#endif
}
}

void TTestWindow::CMDIS80()
{
// This displays the pointer table area of memory to see if the board is
active
// and set up correctly.

WORD data[24];
char msgbuf[240];
char temp[7];
int i;

strcpy(msgbuf,"");
strcpy(temp,"");

for(i=0;i<24;i++)
{
data[i] = MGR(0x80+i);
sprintf(temp,"%04X  ",data[i]);
strcat(msgbuf,temp);
if((i+1)%8 ==0)
strcat(msgbuf,"\n");
}
::MessageBox(HWindow,msgbuf,"Monitor values from 80h to 98h",MB_OK);

strcpy(msgbuf,"");
strcpy(temp,"");

for(i=0;i<24;i++)
{
data[i] = BGR(0x80+i);
sprintf(temp,"%04X  ",data[i]);
strcat(msgbuf,temp);
if((i+1)%8 ==0)
strcat(msgbuf,"\n");
}
::MessageBox(HWindow,msgbuf,"Mother Board values from 80h to 98h",MB_OK);

strcpy(msgbuf,"");
strcpy(temp,"");
for(i=0;i<24;i++)
{
data[i] = BGR(0x40+i);
```

```
sprintf(temp,"%04X  ",data[i]);
strcat(msgbuf,temp);
if((i+1)%8 ==0)
strcat(msgbuf,"\n");
}
::MessageBox(HWindow,msgbuf,"Mother Board values from 40h to 58h",MB_OK);
}

void TTestWindow::CMF2040()
{
// This functions fills the beginning of the board monitor memory area with
// consecutive numbers to test the MPR function.

int i;

for(i=0;i<24;i++)
MPR(0x2880+i,i);

}
void TTestWindow::CMD2040()
{
// this function reads the monitor memory area for verifying the previous
// function.

WORD data[24];
char msgbuf[240];
char temp[7];
int i;

strcpy(msgbuf,"");
strcpy(temp,"");

for(i=0;i<24;i++)
{
data[i] = MGR(0x2880+i);
sprintf(temp,"%04X  ",data[i]);
strcat(msgbuf,temp);
if((i+1)%8 ==0)
strcat(msgbuf,"\n");
}
::MessageBox(HWindow,msgbuf,"Monitor values from 2080h to 2098h",MB_OK);

}
void TTestWindow::CMZERO()
{
#ifdef TYPEADI3
SetMonitorClock(0,0,0,0);
#else
SetPC3Clock(0,0,0,0);
#endif
}

void TTestWindow::CMRT1()
{
// This function returns any new data from RT 1 SA-1 Recieve. To use this
// function the NoPass flag must be false and the initialize board function
// must have been run, and RT1 Receive SA1 should be set up with at least 3
// data buffers.

WORD data[34];
char msgbuf[240];
char temp[7];
int i;

strcpy(msgbuf,"");
strcpy(temp,"");
```

```
GetRtData(1,1,0,data);
sprintf(msgbuf,"Command = %04X WC = %02d\n",data[0],data[1]);
for(i=2;i<34;i++)
{
sprintf(temp,"%04X  ",data[i]);
strcat(msgbuf,temp);
if((i-1)%8 == 0)
strcat(msgbuf,"\n");
}
::MessageBox(HWindow,msgbuf,"1-R-1 Data",MB_OK);

}

void TTestWindow::CMRT1TRANS()
{
// This function Fills RT1-T-1 with new data to test the PutRtData function.
// To use this function the NoPass flag must be false and the initialize
// board function must have been run, and RT1 Transmit SA1 should be set up
// with at least 3 data buffers.

WORD data[34];
int i,j;

::MessageBox(HWindow,"Begin Filling RT1-T-SA1 Using RtPutData","1-R-1 Da-
ta",MB_OK);
for(i=0;i<10;i++)
{
for(j=0;j<32;j++)
data[j] = i;
PutRtData(1,1,data);
// stall for a while
for(j=0;j<10000;j++);
}
::MessageBox(HWindow,"End RTPutData Test","1-R-1 Data",MB_OK);

}

void TTestWindow::CMCLOSE()
{
if (BC_Running)
Stop_BC();

if (BCLoaded)
CloseBCFile();

if(NoPass)
StopUC();

/
***************************************************************************
***********/
// The call to FreeLibrary unloads the dll.
if(!BadLib)
{
CloseMonitor();
FreeLibrary(PassLib);
}
/
***************************************************************************
***********/

if (fileopen)
_lclose(testfile);
this->CloseWindow();
}
```

```
void TTestWindow::CMINITL()
{
// This function sets up data logging. The logging functions were designed to
// run independantly from the PASS so NoPass should be true. The logging
functions
// will run with or without the PASS software. The board needs to be ini-
tialized
// before running this function.

testfile = OpenFile("test.arc",&AnOfstruct,OF_CREATE | OF_WRITE);
if(testfile ==-1)
::MessageBox(HWindow,"Unable to Open Test file!","File Error",MB_OK);
else
// This function loads the interrupt service routine and sets a filesize
// for archiving of 240000 bytes (about four data buffers )

InitializeLog(testfile,240000);

}

void TTestWindow::CMSTARTL()
{
// Start data logging to file test

StartMonitor();
}

void TTestWindow::CMSTOPL()
{

// stop monitoring.
StopMonitor();
}


void TTestWindow::CMSTARTSS()
{
WORD tg_loc, err_only, mon_spurious;
// Start snapshot monitor
tg_loc = 0;
err_only = FALSE;
mon_spurious = TRUE;
StartSnapshot(tg_loc, err_only, mon_spurious);
}

void TTestWindow::CMSTOPSS()
{

// stop monitoring.
StopSnapshot();
}

void TTestWindow::CMCONVERTBA()
{
// Converts file "test" to ascii in file test.asc
ConvertBA("test.arc","test.txt");
}

void TTestWindow::CMCONVERTAB()
{
WORD ret_word;
char outstr[80];
// Converts file "test" to ascii in file test.asc
ret_word = ConvertAB("test.txt","test2.arc");
```

```
sprintf(outstr,"ConvertAB return word = %hu",ret_word);
::MessageBox(0,outstr,"Convert ASCII to Binary Return",MB_OK);


}

void TTestWindow::CMCONVERT23()
{
WORD ret_word;
char outstr[80];
// Converts file "test" to ascii in file test.asc
ret_word = Convert23("test.bam","test.ba3");
sprintf(outstr,"Convert23 return word = %hu",ret_word);
::MessageBox(0,outstr,"Convert BAM to BA3 Return",MB_OK);


}
void TTestWindow::CMCONVERT32()
{
WORD ret_word;
char outstr[80];
// Converts file "test" to ascii in file test.asc
ret_word = Convert32("test.ba3","test.bam");
sprintf(outstr,"Convert32 return word = %hu",ret_word);
::MessageBox(0,outstr,"Convert BA3 to BAM Return",MB_OK);


}

void TTestWindow::CMBCLOAD()
{
OpenBCFile("test.bc", 15);

BCLoaded = (Load_BC( TRUE, 0) == LOAD_OK);
if(!BCLoaded)
{
::MessageBox(HWindow,"Unable to Load BC File!","File Error",MB_OK);
CloseBCFile();
}
}


void TTestWindow::CMBCSTART()
{
if (BCLoaded)
{
BC_Running = Start_BC( );
if (!BC_Running)
::MessageBox( HWindow,"Unable to Start BC List!", "BC Error",MB_OK);
}
else
::MessageBox( HWindow,"Load BC File before Starting List Processing!",
"File Error",MB_OK);
}

void TTestWindow::CMBCSTOP()
{
if (BC_Running)
Stop_BC( );
}
void TTestWindow::CMBCCLOSE()
{
if (BC_Running)
Stop_BC();
if (BCLoaded)
CloseBCFile();
```

```
}

void TTestWindow::CMBCREAD()
{
WORD mess[180];
FILE *out;
WORD readerr;
int i;

// this function uses the ReadBCMess to read in all the bc messgaes in
// the file test.bc and writes the messages out to file out.txt. It does not
// need the boards to be initialized.

out = fopen("bcout.txt", "w");
OpenBCFile("test.bc", 15);
while((readerr = ReadBCMess(mess)) != END_OF_FILE )
{
if (readerr != MESS_OK)
fprintf(out,"Error = %hu\n",readerr);
else
{
switch(mess[0])
{
case BC_Type_FR :
fprintf(out,"Frame\t%04X\t%04X\n\n",mess[Cmd1Er2], mess[Cmd1Er1]);
break;

case BC_Type_ext:
fprintf(out,"Wait for External\n\n");
break;

case BC_Type_Md :
fprintf(out,"Mode\t%04X\t%04X\t%04X\t%04X\n",
mess[Cmd1Er1], mess[Cmd1Er2], mess[Cmd1], mess[IMG]);
//look for data words
if (mess[DataEr1] != MT_EndMsg)
fprintf(out,"Data\t%04X\t%04X\t%04X\n\n",mess[DataEr1], mess[DataEr2],
mess[Data1]);
else
fprintf(out,"\n");
break;

case BC_Type_RB :
fprintf(out,"RT-BC\t%04X\t%04X\t%04X\t%04X\n\n",
mess[Cmd1Er1], mess[Cmd1Er2], mess[Cmd1], mess[IMG]);
break;

case BC_Type_RR :
fprintf(out,"RT-RT1\t%04X\t%04X\t%04X\t%04X\n",
mess[Cmd1Er1], mess[Cmd1Er2], mess[Cmd1], mess[IMG]);
fprintf(out,"RT-RT2\t%04X\t%04X\t%04X\n\n",mess[DataEr1], mess[DataEr2],
mess[Data1]);
break;

case BC_Type_BR : case BC_Type_BRxtra :
if( mess[0] == BC_Type_BR)
fprintf(out,"BC-RT\t%04X\t%04X\t%04X\t%04X\n",
mess[Cmd1Er1], mess[Cmd1Er2], mess[Cmd1], mess[IMG]);
else
fprintf(out,"Buffer\t%04X\t%04X\t%04X\t%04X\n",
mess[Cmd1Er1], mess[Cmd1Er2], mess[Cmd1], mess[IMG]);

i = 0;
fprintf(out,"Data");
while((mess[DataEr1 +(i*3)] != MT_EndMsg) && ( i < 33))
{
```

```
fprintf(out,"\t%04X\t%04X\t%04X",
mess[DataEr1 +(i*3)], mess[DataEr2+(i*3)], mess[Data1+(i*3)]);
i++;
if((i>0)&&((i%3) == 0) && (mess[DataEr1 +(i*3)] != MT_EndMsg))
fprintf(out,"\nData");
}
fprintf(out,"\n\n");
break;
default :
fprintf(out,"Unknown Message Type\n\n");
break;
} //end switch
}  // end else message !ok
}   // end while not EOF

CloseBCFile();
fclose(out);
}

void TTestWindow::CMBCEDIT()
{
/* This demonstrates the use of the calls needed to edit a
bc-rt message data buffer while the BC list is running .
Particular care must be taken to insure that the new buffer
is exactly the same size as the replaced buffer to insure
that the BC list processing is not disrupted. The BCCPTR (0x85)
on the mother board can be monitored to reduce the chance that
a buffer will modified during transmission. This example assumes that
message 2 in a BC list (the first message after the Frame) is a
BC-RT and it has 32 words and one error. */

WORD msg = 2, msgadr, msgofs;
WORD buf = 1, bufadr, bufofs;
WORD PassBuf[100];
WORD databuf[32];
WORD outbuf[100];
WORD wrdcnt;
char msgbuf[240];
char temp[7];
ERR_ENTRY errentry;
char outstr[180];
int i;

// initialize test data array
for(i=0;i<32;i++)
databuf[i] = 0xDEAD;

// make a single error entry for word 2(zero indexed )
// the 0xb3 corresponds to a parity error
errentry.index = 1;
errentry.err1 = 0;
errentry.err2 = 0x00B3;
errentry.next = NULL;

if(!BC_Start_Edit())
{
::MessageBox(HWindow,"Error Initializing BC Edit Module in PASSLIB DLL!",
"File Error",MB_ICONEXCLAMATION);
return;
}
else
/* find message and buffer address and offset and load in new
data buffer */
if ( BC_Find_Message( msg, &msgadr, &msgofs))
{
if ( BC_Find_Buffer( buf, msgadr, msgofs, &bufadr, &bufofs))
```

```
{
sprintf(outstr,"Message %d Buffer %d at Address = %0X Offset = %0X",
msg,buf,bufadr,bufofs);

::MessageBox(HWindow,outstr,"Search Result",MB_OK);


// Get the current buffer value and display it
Get_BC_Buffer(bufadr, bufofs, outbuf,&wrdcnt);
strcpy(msgbuf,"");
strcpy(temp,"");

for(i=0;i<32;i++)
{
sprintf(temp,"%04X  ",outbuf[(i*3)+2]);
strcat(msgbuf,temp);
if((i+1)%8 ==0)
strcat(msgbuf,"\n");
}
::MessageBox(HWindow,msgbuf,"BC-RT Data Buffer from Msg 2",MB_OK);

// Convert the test databuf and errentry to PASS 3 word format
Convert_BC_Buffer( databuf, PassBuf, NULL, 32);

// put the buffer in the found address. This function compares the
// size of the new and existing buffers and returns false if the
// buffer are not exactly =.
if (!Put_BC_Buffer( bufadr, bufofs, PassBuf))
::MessageBox(HWindow,"Buffers are not the same size!",
"File Error",MB_ICONEXCLAMATION);

BC_End_Edit();
return;
}
else
{
::MessageBox(HWindow,"Buffer Not Found!",
"File Error",MB_ICONEXCLAMATION);
BC_End_Edit();
return;
}
}
else
{
::MessageBox(HWindow,"Message Not Found!",
"File Error",MB_ICONEXCLAMATION);
BC_End_Edit();
return;
}
}



void TTestWindow::CMRTREAD()
{
FILE *out;
WORD rt_return, sa_return;
int i;
WORD RT, SA;
char rtlabel[15];
WORD sabuf[110];
WORD rtdef[65];
BOOL lastrt;
BOOL emulate;

// This function uses the GetNextRTDEF and GetNextSABuffer to read in
// an ASCII RT setup file
out = fopen("rtout.txt", "w");
OpenRTFile("test.rt", 15);
// The next call will load the RT setup into the PASS
/* if(!LoadRTSetup())
::MessageBox(HWindow,"Error Loading test.rt onto PASS!",
"File Error",MB_ICONEXCLAMATION);

*/
do
{
rt_return = GetNextRTDEF( &RT, rtdef , rtlabel);
if ( rt_return == DEF_OK )
{
fprintf( out,"RT = %d; Label = %s; MODE = %04X; Control = %04X;\n",
RT, rtlabel, rtdef[RT_MODE],rtdef[RT_CTL]);
if((rtdef[RT_MODE] & 0x2) == 0)
emulate = TRUE;
else
emulate = FALSE;

if(( rtdef[RT_BIT0] != RT_RspGap) || ( rtdef[RT_BIT1] != RT_Wrd1) ||
( rtdef[RT_BIT2] != RT_DtaWrd2) || ( rtdef[RT_BIT3] != RT_Wrd3))
fprintf( out, "\tBIT values\t%04X\t%04X\t%04X\t%04X\n",
rtdef[RT_BIT0],rtdef[RT_BIT1], rtdef[RT_BIT2], rtdef[RT_BIT3]);

if(( rtdef[RT_LCD0] != RT_RspGap) || ( rtdef[RT_LCD1] != RT_Wrd1) ||
( rtdef[RT_LCD2] != RT_DtaWrd2) || ( rtdef[RT_LCD3] != RT_Wrd3))
fprintf( out, "\tLast Command\t%04X\t%04X\t%04X\t%04X\n",
rtdef[RT_LCD0], rtdef[RT_LCD1], rtdef[RT_LCD2], rtdef[RT_LCD3]);

if(( rtdef[RT_TVW0] != RT_RspGap) || ( rtdef[RT_TVW1] != RT_Wrd1) ||
( rtdef[RT_TVW2] != RT_DtaWrd2) || ( rtdef[RT_TVW3] != RT_Wrd3))
fprintf( out, "\tVector Word\t%04X\t%04X\t%04X\t%04X\n",
rtdef[RT_TVW0], rtdef[RT_TVW1], rtdef[RT_TVW2], rtdef[RT_TVW3]);

if(( rtdef[RT_LSW0] != RT_RspGap) || ( rtdef[RT_LSW1] != RT_Wrd1) ||
( rtdef[RT_LSW2] != RT_CmdWrd2) || ( rtdef[RT_LSW3] != (RT << 11)))
fprintf( out, "\tLast Status\t%04X\t%04X\t%04X\t%04X\n",
rtdef[RT_LSW0], rtdef[RT_LSW1],  rtdef[RT_LSW2], rtdef[RT_LSW3]);

if(( rtdef[RT_MD0] != RT_RspGap) || ( rtdef[RT_MD1] != RT_Wrd1) ||
( rtdef[RT_MD2] != RT_CmdWrd2) || ( rtdef[RT_MD3] != (RT << 11)))
fprintf( out, "\tAll Modes\t%04X\t%04X\t%04X\t%04X\n",
rtdef[RT_MD0], rtdef[RT_MD1], rtdef[RT_MD2], rtdef[RT_MD3]);

do
{
sa_return = GetNextSABuffer(&SA, sabuf);

if (sa_return == DEF_OK)
{
if (SA < 32)
{
fprintf( out,"SA = %d R Gap = %04X Status = %04X %04X %04X\n",
SA, sabuf[RSGap], sabuf[RSTS0], sabuf[RSTS1], sabuf[RSTS2]);
}
else
{
fprintf( out,"SA = %d T Gap = %04X Word Count = %d Status = %04X %04X %04X\n",
SA-32, sabuf[RSGap], (int)sabuf[WrdCnt],
sabuf[RSTS0], sabuf[RSTS1], sabuf[RSTS2]);
if (emulate)
{
fprintf(out,"Data");
for(i=0;i<33;i++)
{
fprintf(out,"\t%04X\t%04X\t%04X",
sabuf[TERR1 +(i*3)], sabuf[TERR2+(i*3)], sabuf[TDATA+(i*3)]);
if((i>0)&&((i+1)%3 == 0) && (i+1 != 33))
fprintf(out,"\nData");
}
fprintf(out,"\n");
}
}
}
else
if((sa_return != SA_END) && (sa_return != RTEND_OF_FILE))
{
fprintf(out, "SA Error = %d\n", sa_return);
sa_return = RTEND_OF_FILE;
}
}
while ((sa_return != SA_END)&& ( sa_return != RTEND_OF_FILE ));
fprintf(out,"\n");
}
else
if(rt_return != RTEND_OF_FILE)
{
fprintf(out, "RT Error = %d\n", rt_return);
rt_return = RTEND_OF_FILE;
}
}
while (( rt_return != RTEND_OF_FILE ) && ( sa_return != RTEND_OF_FILE ) &&
( rt_return != RTFILE_ERR));
CloseRTFile();
fclose(out);
}

void TTestWindow::CMRTLOAD()
{
OpenRTFile("test.rt", 15);
// The next call will load the RT setup into the PASS
if(!LoadRTSetup())
::MessageBox(HWindow,"Error Loading test.rt onto PASS!",
"File Error",MB_ICONEXCLAMATION);
CloseRTFile();
}



void TTestWindow::CMTOOLREAD()
{
FILE *out;
WORD type;
WORD ret_word= READ_OK, outlevel, mtimeout, btimeout, itype, ibase;
WORD noeom, tloc;
int ifreq,i,j;
long ioffset;
TNAMES modenames;
WORD   modetype[32];
TFILTERS filters;
TTRIGGERS triggers;
float bdis_time, mdis_time;
BOOL def_modes;
char charstr[2];
char outstr[20];
WORD simple, err_only;


// this function uses the GetNextType function to find the various
// parts of a misc. Tool file and recover the data.
```

```
out = fopen("toolout.txt", "w");
OpenToolFile("test.tl", 15, 10);


fprintf(out, "    Tool Data from File : Test.tl \n\n");
while (ret_word != TEND_OF_FILE)
{
ret_word = TGetNextType(&type);
if(ret_word == READ_OK)
{
switch(type)
{
case T_OLEVEL :
ret_word = TGetOutLevel(&outlevel);
if(ret_word == READ_OK)
fprintf(out, "Output Level = %d\n\n", outlevel);
else
fprintf(out, "Output Level Read Error = %d\n\n", ret_word);
break;

case T_TIMEOUT :
ret_word = TGetTimeOut( &btimeout, &mtimeout );
if(ret_word == READ_OK)
{
if (btimeout == 0xffff)
bdis_time = 7.0;
else
bdis_time = ((-1 *((int)btimeout)/15.0) + 2.0);

if (mtimeout == 0xffff)
mdis_time = 7.0;
else
mdis_time = ((-1 *((int)mtimeout)/10.0) + 2.0);

fprintf( out, "Status Responce Timeout = %4.1f  %4.1f\n\n",
bdis_time, mdis_time);
}
else
fprintf(out, "Status Responce Timeout Error = %d\n\n", ret_word);
break;

case T_MODEA :
ret_word = TGetModes( &(modenames[0][0]), modetype);
if(ret_word == READ_OK)
{
def_modes = TRUE;
i= 0;
while((def_modes) && (i<= 31))
{
if((modenames[i][0]!='\0')||( modetype[i] != MRECEIVE))
def_modes = FALSE;
i++;
}

/* if any mode names are non-empty or set as transmit then
write out all non-default modes */
if (!def_modes)
{
fprintf(out, "1553 MODE \n");
fprintf(out, " Mode T/R    Name\n");
for(i= 0;i<=31;i++)
{
if((modenames[i][0]!='\0')||( modetype[i] != MRECEIVE))
{
if( modetype[i] == MTRANSMIT)
strcpy(charstr,"T");
else
strcpy(charstr,"R");
fprintf(out, " %02d    %s    %s \n",i,charstr,modenames[i]);
}
}
fprintf(out, "\n");
}
else
fprintf(out," Default 1553A Mode Code Setup \n\n");
}
else
fprintf(out, "1553A Mode Code Setup Read Error = %d\n\n", ret_word);
break;

case T_IRIG :
ret_word = TGetIrig(&itype, &ibase, &ifreq, &ioffset);
if(ret_word == READ_OK)
{
if( itype == TDOS)
strcpy(charstr,"D");
else
if( itype == TUSER)
strcpy(charstr,"U");
else
strcpy(charstr,"I");

fprintf(out,
"IRIG Setup \n   Source : %s, Base Address : %X  Frequency : %d Correction
: %ld\n\n",
charstr,ibase,ifreq,ioffset);
}
else
fprintf(out, "IRIG Setup Read Error = %d\n\n", ret_word);
break;

case T_FILTER :
ret_word =  TGetFilters(&err_only, &(filters[0][0]));
if(ret_word == READ_OK)
{
if(err_only)
strcpy(outstr,"On");
else
strcpy(outstr,"Off");
fprintf(out, " Filter Setup \n ");
fprintf(out,"   Errors Only = %s\n", outstr);
fprintf(out,"RT   Word1  Word2  Word3  Word4\n");
for(i=0;i<32;i++)
fprintf(out," %02d   %04X   %04X   %04X   %04X\n",
i,filters[i][0], filters[i][1], filters[i][2], filters[i][3]);
fprintf( out, "\n");
}
else
fprintf(out, "Filter Setup Read Error = %d\n\n", ret_word);
break;


case T_TRIGGER :
ret_word =  TGetTriggers( &noeom, &tloc, &simple,&(triggers[0][0]));
if(ret_word == READ_OK)
{
if( tloc == TSTART)
strcpy(charstr,"S");
else
if( tloc == TMIDDLE)
strcpy(charstr,"M");
```

```
else
strcpy(charstr,"E");

if(simple)
strcpy(outstr,"SIMPLE");
else
strcpy(outstr,"COMPLEX");

fprintf(out," Trigger Setup \n");
fprintf(out,"  EOM  = %04X;  Location = %s    %s\n", noeom,charstr,outstr);
fprintf(out," Block  Exp.  Mask  Value Count Goto1 Goto2 Goto3 Flag\n");
for(i=0;i<16;i++)
fprintf(out,"  %02d    %04X  %04X  %04X  %04X  %04X  %04X  %04X  %04X\n",
i, triggers[i][0], triggers[i][1], triggers[i][2], triggers[i][3],
triggers[i][4], triggers[i][5], triggers[i][6], triggers[i][7]);
fprintf( out, "\n");
}
else
fprintf(out, "Trigger Setup Read Error = %d\n\n", ret_word);
break;

case T_END :
fprintf(out, "End of File \n");
ret_word = TEND_OF_FILE;
break;

default :
ret_word = TEND_OF_FILE;
fprintf(out, "GetType Failure\n");
break;
} //end case
} // end if ret_word == Read_ok
} // while not end of file
if(ret_word == TEND_OF_FILE)
fprintf(out,"End of File\n");
fclose(out);
CloseToolFile();
}

void TTestWindow::CMTOOLLOAD()
{
// The next call will load the Tool setup into the PASS
if(!LoadToolFile("test.tl", 15, 10))
::MessageBox(HWindow,"Error Loading test.tl onto PASS!",
"File Error",MB_ICONEXCLAMATION);
}
void TPASSTest::InitMainWindow()
{
MainWindow = new TTestWindow(NULL, "Create a DLL Window");
}

int PASCAL WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
   LPSTR lpCmd, int nCmdShow)
{
TPASSTest PASSTestApp("PASSTEST", hInstance, hPrevInstance,
lpCmd, nCmdShow);
PASSTestApp.Run();
return(PASSTestApp.Status);
}

DEFINE_RESPONSE_TABLE1( TTestWindow, TFrameWindow )
EV_COMMAND( 101, CMINIT ),
EV_COMMAND( 102, CMDIS80 ),
EV_COMMAND( 103, CMF2040 ),
EV_COMMAND( 104, CMD2040 ),
EV_COMMAND( 106, CMZERO ),
EV_COMMAND( 105, CMCLOSE ),
EV_COMMAND( 201, CMINITL ),
EV_COMMAND( 202, CMSTARTL ),
EV_COMMAND( 203, CMSTOPL ),
EV_COMMAND( 204, CMCONVERTBA ),
EV_COMMAND( 205, CMCONVERTAB ),
EV_COMMAND( 206, CMCONVERT23 ),
EV_COMMAND( 207, CMCONVERT32 ),
EV_COMMAND( 301, CMBCLOAD),
EV_COMMAND( 302, CMBCSTART),
EV_COMMAND( 303, CMBCSTOP),
EV_COMMAND( 304, CMBCCLOSE),
EV_COMMAND( 305, CMBCREAD),
EV_COMMAND( 601, CMBCEDIT),
EV_COMMAND( 400, CMRTREAD),
EV_COMMAND( 401, CMRTLOAD),
EV_COMMAND( 402, CMRT1),
EV_COMMAND( 403, CMRT1TRANS),
EV_COMMAND( 501, CMSTARTSS),
EV_COMMAND( 502, CMSTOPSS),
EV_COMMAND( 701, CMTOOLREAD),
EV_COMMAND( 702, CMTOOLLOAD),
END_RESPONSE_TABLE;
```